تبسيط البرمجة

تأليف

Robertson, Lesley



ترجمة

المهندس / علي يوسف علي

خوارزم للنشر والتوزيع و الكمبيوتر

تبسيط البرمجة

تأليف

Robertson, Lesley

ترجمة

المهندس / على يوسف علي


خوارزم للنشر والتوزيع و الكمبيوتر

يطلب من

مكتبة علاء الدين

العنوان : 63 شارع صفية زغلول - محطة الرمل - الإسكندرية

☎ 4836186

مكتبة الصفدى

العنوان : شارع سعد الله الجابرى -مقابل البريد - دمشق - سوريا

☎ 2218016

بسم الله الرحمن الرحيم

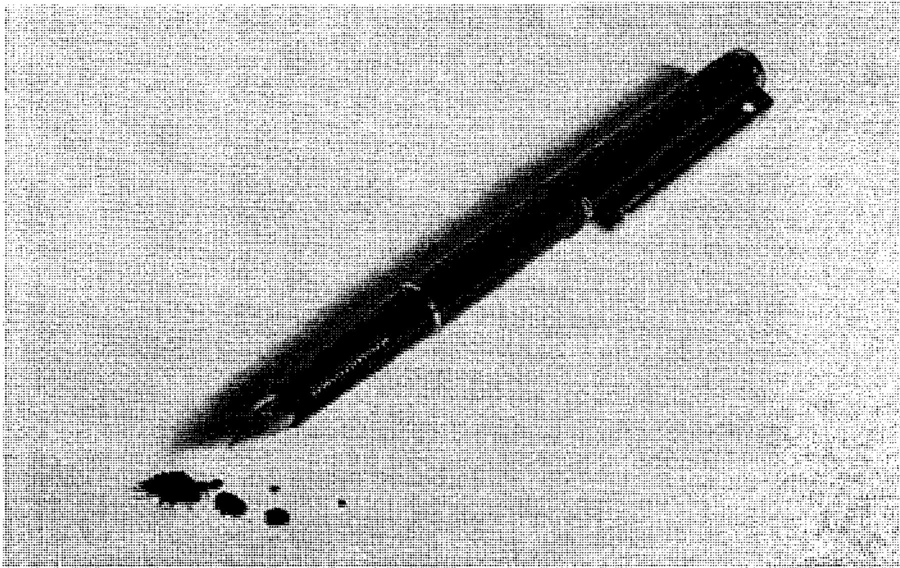
إهداء ...

إلى العالم العربي المسلم " محمد بن جعفر الخوارزمي " أبو الرياضيات الذي جعـــل من معارف الجبر الممزقة علماً لأول مرة ، وأدرك قيمة " الصفر " في الحساب ... إليه وإلى إخوانه من علمائنا الأفذاذ أصحاب الفضل في استمرار مســـيرة العلـــم وإرساء مبادئه .. نهدي هذا الكتاب .

خوارزم للنشر والتوزيع

فهرس

الفصل الأول تصميم البرامج



الأهداف:

- ➢ شرح خطوات عملية وضع البرامج
- ➢ مفهوم البرمجة المهيكلة
- ➢ التعريف بالخوارزم واللغة شبه البرمجية pseudocode

الخطوط العامة :

1-1 خطوات وضع البرامج

1-2 البرمجة المهيكلة

1-3 مقدمة عن الخوارزم واللغة شبه البرمجية

1-4 ملخص الفصل

1-1 خطوات وضع البرامج

البرمجة فن، فالكثير يعتقدون أن المبرمج محتاج أن يكون متخصصا في الرياضيات، وذو عقلية مؤهلة للتعامل مع الأرقام والمعلومات، ومستعد لأن يظل ساعات طوال جالسا إلى جهازه، يكتب البرامج ويصححها. إلا أنه في الواقع لو أن المبرمج أمد بالوسائل المناسبة، وأسلوبا منهجيا يسير عليه، فإن البرمجة تصبح متاحة للجميع. إنها مهمة يجب تنفيذها، ولكنها في نفس الوقت تحمل الكثير من التحفيز والشعور بالرضا عن النفس.

ويمكن تعريف عملية البرمجة على أنها وضع تصور عام لتنفيذ مهمة تم توصيفها، ثم وضع الخطوات التنفيذية لهذا الحل على صورة تعليمات متتالية حينما توجه لمكونات الحاسوب المادية تؤدي للنتيجة المطلوبة. ويتمثل الجانب الإبداعي الخلاق من البرمجة في أول جزء من التعريف، ألا وهو وضع الالتصور العام للحلول للمشاكل التي يتم مواجهتها. ومع ذلك فغالبا ما يتم تجاهل هذه الخطوة، بالقفز فورا إلى مرحلة صياغة البرنامج دون وضع تصميم واضح لما يكون عليه الحل، وغالبا ما يترتب على ذلك الكثير من الأخطاء المنطقية logical errors التي تتطلب جهدا كبيرا في اكتشافها وتصحيحها. أما المبرمج الأكثر خبرة فيضع في البداية تصميما للحل، ثم يختبره مكتبيا، وعند الاطمئنان على صحته يترجمه إلى أوامر برمجية باللغة التي اختارها.

وتتمثل خطوات وضع الحل لمشكلة ما في سبع نقاط نعرض إطارها العام فيما يلي:

1- حل المشكلة

تتطلب هذه الخطوة قراءة متأنية للمشكلة إلى أن يتم استيعابها تماما ومعرفة المطلوب بكل وضوح، وحتى يمكن القيام بهذه الخطوة الأولية بنجاح، تقسم المسألة إلى:

- المدخلات
- المخرجات
- الخطوات المطلوبة للوصول للمخرجات

ونحبذ في ذلك الاستعانة بجدول تحليلي للتعرف على المشكلة كما سنقدمه في الفصل الثالث، إذ يساعد ذلك على فصل هذه العناصر الثلاثة والتعرف عليها.

2- ضع تصورا عاما للحل

بمجرد تحديد المشكلة بوضوح، قد ترى تجزئتها إلى خطوات أو مهام أبسط، وتضع بذلك تصورا عاما للحل. هذا التصور المبدئي هو عادة مجرد خطوط عامة للحل يمكن أن تحتوي على في حالة البرامج الحاسوبية على:

- خطوات المعالجة الأساسية
- المهام الفرعية الرئيسية (إن وجدت)
- الهياكل الرئيسية للتحكم في سير التنفيذ (كالدوارات loops أو القرارات decisions)
- المتغيرات الأساسية وهياكل البيانات
- المنطق العام للبرنامج

قد يتضمن التصور العام للحل أيضا خريطة هيكلية على شكل تدرج هرمي، ونناقش في الفصلين الثاني والثالث خطوات وضع هذا التصور العام.

3 _ ضع التصور العام على هيئة خوارزم

يمكن تطوير التصور العام الذي وضع في الخطوة السابقة إلى خوارزم، أي مجموعة من الخطوات المحددة التي تصف بدقة المهام المنفذة وترتيب تنفيذها. ويستخدم هذا الكتاب أسلوب اللغة شبه برمجية pseudocode، تتميز بكونها قريبة من اللغة المهيكلة (متفق عليه لوصف النظم خلال عملية تحليل وتصميم النظم، راجع كتاب "تحليل وتصميم نظم المعلومات" ترجمة المهندس علي يوسف علي وإصدار دار خوارزم-المترجم structured language لتمثيل الحل، وأيضا لتمثيل تكنيكات البرمجة المهيكلة structured programming techniques. كما نقدم أيضا أشكال ناسي-شنايدر لمن يفضلون التمثيل التصويري للخوارزمات. وسوف نناقش

باستفاضة في الفصول من 2 إلى 6 استخدام اللغة شبه البرمجية و "نظرية الهيكلة Structure Theorem" في وضع الخوارزمات.

4- اختبر صلاحية الخوارزم

هذه الخطوة من أهم الخطوات في وضع البرامج ورغم ذلك فكثيرا ما يتم تجاهلها. وأهمية هذه الخطوة هي اكتشاف الأخطاء المنطقية في الخوارزمات مبكرا. وتستخدم بيانات اختبارية في عملية التشغيل التجريبي لتسير مع الخوارزم خطوة فخطوة، للتأكد من أن التعليمات الخاصة بالخطوات ستؤدي بالفعل لما هو مطلوب. وهنا يسير المبرمج مع خطوات الخوارزم، بالضبط كما سيفعل الحاسب عند التنفيذ، مسجلا التغير في قيم المتغيرات الأساسية. ويقدم الفصل الثالث جداول وخوارزمات اختبار مقترحة، والعديد من الأمثلة التوضيحية.

5- صغ الخوارزم بلغة برمجية

وليس لك أن تنشط لصياغة برنامجك في لغته البرمجية إلا بعد الوفاء بكل متطلبات التصميم كما عرضناها في الخطوات الأربع السابقة.

6- شغل برنامجك على الحاسوب

هذه الخطوة الحاسمة، والمثيرة في نفس الوقت، وبافتراض أنك قد بذلت العناية الكافية في التصميم، فإن ساعات الإحباط لوجود الأخطاء سوف تنكمش لأقل مستواها. وقد يتطلب الأمر تكرار هذه الخطوة عدة مرات قبل الوصول للرضا الكامل عن البرنامج.

7- وثق البرنامج

ليس من اللائق أن توضع خطوة توثيق البرنامج كخطوة أخيرة في تتابع وضع البرامج، حيث إنها في الواقع إجراء يسير مع كافة الخطوات السابقة.

والتوثيق نوعان، خارجي؛ يتمثل في الخرائط الهيكلية، وخوارزمات الحلول، ونتائج الاختبارات، وداخلي؛ وهو ما يضعه المبرمج في صياغة الكود نفسه.

تطوير البرنامج:

بعد أن ينطلق البرنامج في التطبيق العملي، تثور الحاجة عادة إلى إجراء تعديلات عليه مع مرور الزمن، وغالبا ما يكون ذلك من قبل مبرمجين لم يشتركوا في وضع الكود الأصلي. ومن ثم فإنه في حالة البرامج التي بذلت فيها العناية الواجبة في التصميم والتوثيق، تكون إجراءات تطوير البرنامج سهلة تلقائية.

2-1 البرمجة المهيكلة:

البرمجة المهيكلة تساعدك على كتابة برامج على أعلى مستوى من الكفاءة، خالية بقدر الإمكان من الأخطاء المنطقية. وقد ظهر مفهوم البرمجة المهيكلة لأول مرة في ورقة وضعها Bohm وJacopini في إيطاليا عام 1964، حيث قدما فكرة تصميم البرامج باستخدام نظرية الهيكلة Structure Theory، مؤسسة على ثلاثة هياكل حاكمة control structures. ومنذ ذلك الحين تطور المفهوم على يد كثيرين مثل Edsger Dijkstra, Niklaus Wirth, Ed Yourdon, Michael Jackson إلى أن تبلور في الصورة الشائعة لمصطلح "البرمجة المهيكلة". هذا المصطلح لم يعد مقصورا على نظرية الهيكلة ذاتها، بل اتسع ليشمل معها التصميم من أعلى إلى أسفل، والتصميم البنائي modular design أو modularization.

أسلوب من أعلى إلى أسفل في وضع البرامج

كان التقليد الشائع هو أن يبدأ المبرمج بمجرد أن يتعرف على المشكلة في صياغة البرنامج، بادئا من بداية المشكلة متدرجا إلى نهايتها. وغالبا ما يجد نفسه قد انشغل بتفاصيل الخطوات، بدلا من أن يتفرغ لوضع تصور لحل شامل للمشكلة. وفي أسلوب التصميم من أعلى إلى أسفل، يكون البدء بوضع تصور شامل للحل، والذي يتفرع تدريجيا إلى خطوات فرعية قد تتفرع بدورها إلى أن نصل إلى آخر مستوى من التفاصيل. معنى ذلك أن البدء في الصياغة الفعلية للبرنامج لا تكون إلا بعد إجراء هذه العملية التي يطلق عليها:

"التحليل الوظيفي functional decomposition". ونتيجة لهذا الأسلوب المنهجي المنظم نحصل على تصميم على أعلى درجة من الدقة لم يكن متاحا من قبل.

التصميم البنائي

تتضمن البرمجة المهيكلة أيضا فكرة التصميم البنائي، والذي يعني جمع المهام في وحدات طبقا لمنطق معين (مثلا، تهدف لمهمة معينة كحساب الضريبة أو طباعة وثيقة). والتصميم البنائي مرتبط ارتباطا وثيقا بأسلوب الوضع من أعلى إلى أسفل، حيث إن التدرج المشار إليه يتكشف عنه المهام الجزئية، والتي تأخذ شكل الوحدات البنائية modules. ويساعد التصميم البنائي الجيد على فهم البرنامج وتتبعه.

نظرية الهيكلة

تعتبر نظرية الهيكلة ثورة في أسلوب البرمجة، بدأت بإلغاء عبارة GOTO الشهيرة، ووضع إطار هيكلي للحل. وتنص النظرية على إمكانية وضع أي برنامج حاسوبي باستخدام ثلاثة هياكل حاكمة أولية، ألا وهي:

- الأوامر المتتابعة sequences
- القرارات (تسمى أيضا: العبارات الشرطية conditional statemets، والاختيار selection)
- الدوارات loops

وسوف يغطى هذا في الفصل الثاني

3-1 مقدمة عن الخوارزم واللغة شبه البرمجية

تتطلب تكنيكات البرمجة المهيكلة أن يكون البرنامج قد تم تصميمه جيدا قبل البدء في كتابته، وعملية التصميم هذه هي التي يتولد عنها الخوارزم.

ما هو الخوارزم؟

يشبه الخوارزم (وصفات) الطهي، فهي تضع الخطوات لتنفيذ مهمة ما [طهي طبق ما- *المخرجات*، بتحديد المواد المطلوبة ومقاديرها-*المدخلات*، وعمليات الطهي-*المعالجة*]. هذه الوصفات يمكن أن توضع بمصطلحات البرمجة لبيان كيفية الوصول إلى مخرجات معينة من مدخلات معينة. ويكتب الخوارزم بلغة بسيطة واضحة. ويجب الالتزام بمبادئ معينة حتى يكون الخوارزم فعالا:

- أن يكون سلسا، دقيقا، لا يشوبه الغموض؛
- يعطي الحل الذي يغطي كافة الاحتمالات؛
- أن يصل لنهاية محددة.

فعلى سبيل المثال، لو أردت أن تعطي شخصا ما تعليمات لإضافة بعض الأرقام باستخدام الآلة الحاسبة، فقد تضع ذلك على شكل الخوارزم التالي:

⇐ شغل الآلة

⇐ امح أية بيانات موجودة

⇐ كرر التعليمات التالية

◊ أدخل الجزء الصحيح

◊ اضرب على العلامة العشرية

◊ أدخل الجزء الكسري

◊ اضرب على علامة +

إلى أن تنتهي من كافة الأرقام

⇐ اكتب الناتج النهائي

⇐ اطفئ الآلة

وتلاحظ أنه في هذا الخوارزم تنفذ الخطوتان الأوليان مرة واحدة، قبل الدخول في العملية التكرارية لإدخال الأرقام. كما تنفذ الخطوتان الأخيرتان أيضا مرة واحدة.

هذا الخوارزم يفي بكل المتطلبات لخوارزم فعال: فهو يورد كافة الخطوات في ترتيبها الصحيح، وبأسلوب غير غامض أو مبهم، إلى أن نصل إلى النهاية المنشودة. كما تلاحظ أن الخطوات التكرارية قد أزيحت للداخل، لتأخذ وضعا مستقلا يظهر طبيعتها الخاصة. وتعتبر الإزاحة للداخل من الوسائل الهامة للغاية في توضيح البرنامج وتوثيقه.

ما هي اللغة شبه البرمجية؟

كان استعمال مخططات التدفق flow charts من الوسائل الأكثر شيوعا في تصوير الخوارزمات، ولكنها كانت معقدة، وغامضة، وصعبة الفهم والتتبع، وتؤدي بالتالي في كثير من الأحيان إلى أكواد مليئة بالأخطاء. وفي المقابل، تسمح اللغة شبه البرمجية بقراءة خطوات الخوارزم باللغة الطبيعية المألوفة، إلا أنها وضعت على هيئة تجعلها تشابه اللغات البرمجية الراقية.

وكثيرا ما كانت المبرمجون يضعون خوارزمات برامجهم بلغة أقرب للغة التي سوف يستخدمونها في وضع البرامج، وهذا الكتاب يحاول وضع لغة شبه البرمجية لكتابة الخوارزمات، لا تتعلق بلغة برمجية معينة، بل هي منبثقة من التقاليد العامة للغات المهيكلة، على الوجه التالي:

1. تكتب التعليمات بلغة بسيطة.
2. تكتب كل عبارة على سطر مستقل.
3. تستخدم الكلمات الحاكمة key word والإزاحة للداخل لكي تميز الهياكل الحاكمة.
4. تكتب كل مجموعة متميزة من التعليمات (بلوك) من أعلى إلى أسفل، بمدخل واحد ومخرج واحد.
5. قد تضم مجموعات التعليمات في وحدة بنائية مميزة باسم مستقل.

وقد اتبع أسلوب صياغة الخوارزمات بأسلوب اللغة شبه البرمجية في هذا الكتاب لكونها تسمح للمبرمج أن يركز على الجانب المنطقي من برنامجه. على أن البديل التصويري المعروف باسم مخططات ناسي-شنايدر قد تمت معالجته في الملحق رقم 1.

4-1 تلخيص الفصل

تم في هذا الفصل عرض خطوات وضع البرامج مع شرح مختصر لها. هذه الخطوات السبعة هي:

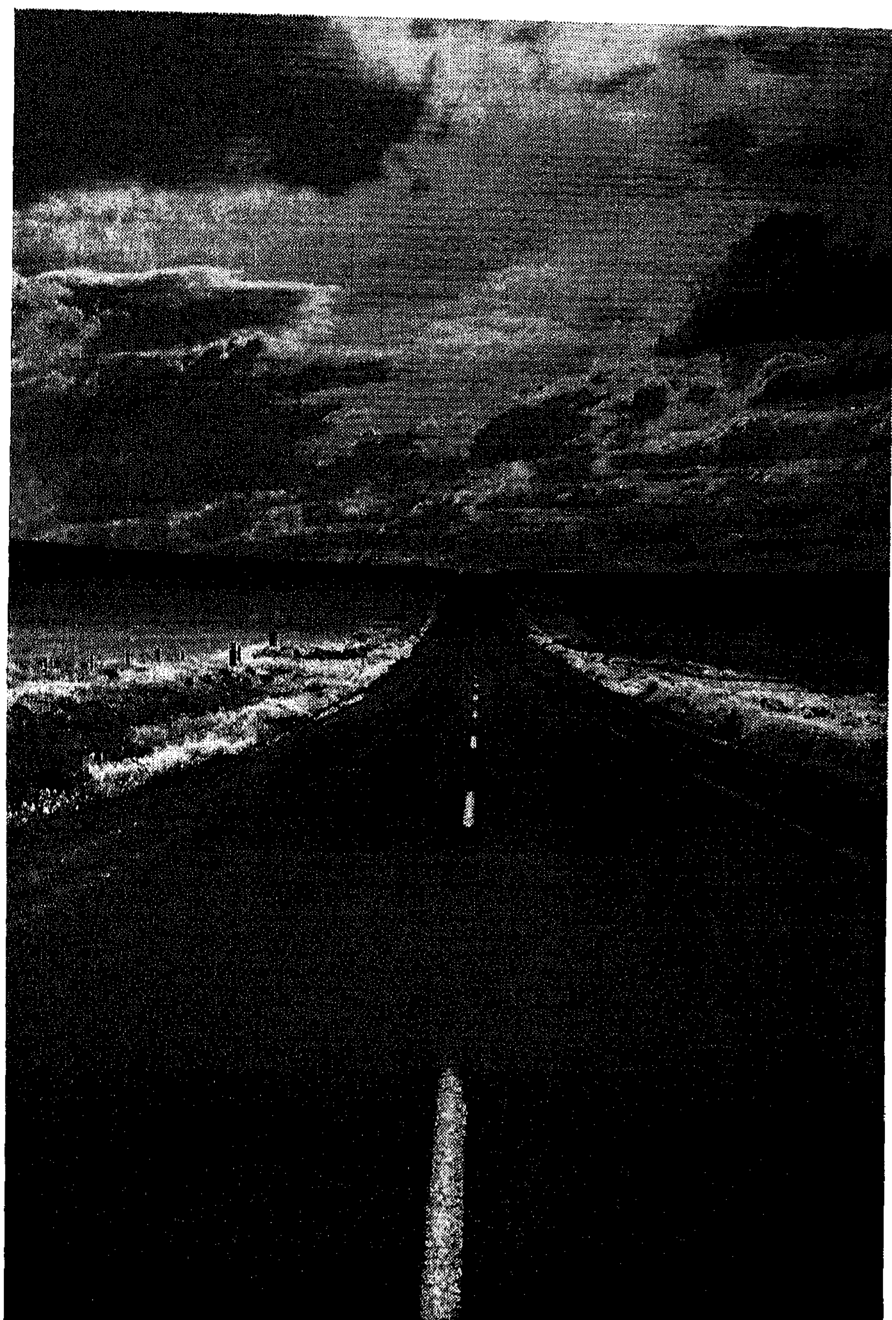
- حدد المشكلة
- ضع تصورا عاما للحل[1]
- ضع خوارزما للحل
- اختبر صلاحية الخوارزم
- صغ البرنامج باللغة المختارة
- شغل البرنامج
- وثق البرنامج.

وقد عرضنا لمفهوم البرمجة المهيكلة، المستخدم لتصميم البرامج كمزيج من ثلاثة مفاهيم: أسلوب من أعلى إلى أسفل، التصميم البنائي، واستخدام نظرية البرمجة المهيكلة.

كما عرف الخوارزم على أنه مجموعة من التعليمات المفصلة، مرتبة وواضحة لا يشوبها الغموض أو الإبهام، تصف عمليات المعالجة اللازمة لإخراج المخرجات من المدخلات. أما اللغة شبه البرمجية فهي هيئة خاصة من اللغة الإنجليزية لوصف الخوارزمات، كما عرضنا لمزاياها ومبادئها الأولية.

---

[1] أسلوب متفق عليه لوصف النظم خلال عملية تحليل وتصميم النظم، راجع كتاب "تحليل وتصميم نظم المعلومات" ترجمة المهندس علي يوسف علي وإصدار دار خوارزم-المترجم



الفصل الثاني اللغة شبه البرمجية



الأهداف:

- التعريف بالكلمات الشائعة والركيزية المستخدمة في اللغة شبه البرمجية.
- تحديد الهياكل الثلاثة الأساسية المتحكمة في سير البرنامج كعناصر أساسية في نظرية الهيكلة.
- توضيح الهياكل الثلاثة الحاكمة.

الخطوط العامة :

2-1 استخدام اللغة شبه البرمجية.

2-2 نظرية الهيكلة

2-3 ملخص الفصل

2-1 استخدام اللغة شبه البرمجية

حينما تضع خوارزما، عليك أن تضع في ذهنك أن السطور التي تكتبها سوف ينفذها حاسوب فيما بعد، ويعني ذلك أنه لو أنك كتبت الخوارزم بكلمات وعبارات تتوافق مع العمليات الأساسية للحاسوب، فستكون عملية الترجمة من الخوارزم إلى البرنامج الفعلي سهلة مباشرة. وسوف نقدم ست عمليات أساسية للحاسوب، ونبين لك الكلمات الركيزية key words والكلمات الشائعة لكل عملية، مصاغة باللغة شبه البرمجية، والتي تقترب كثيرا من اللغة الإنجليزية الطبيعية، وبإزاحات توحي بهيكل حاكم معين.

العمليات الست الأساسية للحاسوب

1- إدخال البيانات:

حينما يراد من الحاسوب أن يتلقى بيان ما، سواء من لوحة الإدخال أو من قرص ممغنط أو أي وسيلة إدخال أخرى، فإن كلمات مثل **READ** و**GET** تكون هي المستخدمة في الخوارزم، وتستخدم كلمة READ عادة حينما يكون الإدخال من سجل أحد الملفات، بينما تستخدم GET للإدخال من لوحة المفاتيح، كما تبين الأمثلة التالية:

```
Read student name
Get system date
Read number_1, number_2
Get tax_code
```

ويستخدم كل مثال أحد فعلي الأمر READ أو GET، متبوعا باسم أو اثنين، يبينان البيانات المطلوب إدخالها. ولا توجد مرحلة على الإطلاق يكون مطلوب فيها تحديد مصدر البيانات، حيث لا تطلب هذه المعلومة إلا عند تشغيل البرنامج.

2- إخراج البيانات:

حينما يراد من البرنامج أن يخرج بيانا ما لأي من أجهزة الإخراج، فإن الكلمات المستخدمة هي Print، حينما يكون الإخراج للطابعة، وWrite حينما يكون البيان موجها لملف مـــا، أمـــا للشاشة فتستخدم كلمات مثل Put, Output, Display. وإليك بعض الأمثلة التطبيقية:

```
Print "Program Completed"
Write customer record to master file
Put out name, address, and postcode
Output total_tax
Display "End of data"
```

وفي كل مثال، يحدد بدقة ما يراد إخراجه، باستخدام الأحرف الصغيرة عادة.

3- إجراء العمليات الحسابية:

يمكن للمبرمج أن يستخدم رموز العمليات الحسابية المعتادة أو الأفعال المعبرة عـــن العمليـــات الحسابية المطلوب إجراؤها، فالتعبيران التاليان مترادفان:

```
Add number to total
total = total + number
```

وتستخدم الرموز التالية في صياغة البرامج عامة:

+ لعملية الجمع

- لعملية الطرح

* لعملية الضرب

/ لعملية القسمة

كما في الأمثلة التالية:

```
Divide total_marks by student_count
sales_tax = cost_price * 0.10
```

كما تضم العمليات المتداخلة في أقواس كما هو متبع في قواعد الرياضيـــات. ويمكـــن أيضـــا استخدام أفعال مثل Compute, Calculate كما في المثال التالي [لأمر تحويـــل درجـــات الحرارة الفهرنهيتية إلى مئوية]:

```
Compute C = (F-32)*5/9
```

4- تخصيص قيمة لبيان

هناك ثلاثة حالات يحتاج فيها إلى تخصيص قيمة لبيان ما:

1. أن تعطى المتغيرات قيما ابتدائية، [غالبا ما تكون قيمة صفرية، وتسمى هـذه العملية الاستهلال initialization] وفي هذه الحالة تستخدم الأفعال Initialize, Set:
2. أن تخصص قيمة كناتج عملية حسابية باستخدام علامة التساوي "="
3. أن تخزن قيمة لاستخدمها في المستقبل، وهنا تستخدم الأفعال Save, Store

وإليك بعض الأمثلة:

```
Initialize total to zero
Set student_count to 0
total_price = cost_price + sales_tax
Store customer_num in last_customer_num
```

5- مقارنة بيانين والاختيار بين إجراءين بديلين

من أهم الإمكانيات المتاحة للمبرمج أن يجعل البرنامج يجري عملية مقارنة بين قيمتين، ثم يتخذ أحد إجراءين طبقا لنتيجة المقارنة، ويستخدم لذلك الكلمات الركيزية **IF, THEN, ELSE**. وتوضع عبارة المقارنة [العبارة الشرطية] بعد الكلمة IF، أما الخيارات [جواب الشرط] فبعد أي من الكلمتين الأخريين، كما يستبين من المثال التالي:

وفي هذا المثال تختبر حالة الطالب من حيث كونه منتظما أو غير منتظم، وطبقا لنتيجة الاختبار تكون **تزايد**[i] increment أي من المتغيرين part_time_count أو full_time_count.

لاحظ استخدام الإزاحة لإعطاء التأثير اللازم لجواب الشرط في الاختيارين، واستخدام عبارة ENDIF للإخبار بانتهاء عملية الاختيار.

```
IF student is part_time
        add 1 to part_time_count
ELSE
        add 1 to full_time_count
ENDIF
```

6- تكرار الأفعال

حينما يراد تكرار عدد من الخطوات، تستخدم الكلمات الركيزية DOWHILE, ENDDO في بداية مجموعة الخطوات التكرارية ونهايتها. ويكون شرط التكرار تابعا للكلمة الأولى، والخطوات ذاتها بعد الشرط، كما في المثال التالي:

```
DOWHILE student_total < 50
        Read student record
        Print student name, address to report
        Add 1 to student_total
ENDDO
```

ومن السهل أن ترى من هذا المثال أن الأوامر المراد تكرارها هي التي تلي عبارة الشرط مباشرة، وتميز بإزاحتها، أما عبارة الشرط فهي التي تلي الكلمة الركيزية DOWHILE، كما تنتهي العملية التكرارية بالكلمة الركيزية ENDDO، وبمجرد أن نصل إلى حالة عدم تحقق الشرط، فإن تنفيذ البرنامج ينتقل إلى الأمر التالي لها مباشرة.

وهناك صياغة أخرى لعبارة الشرط تستخدم العبارة الركيزية WHILE.. DO على الصورة التالية:

```
WHILE student_total < 50  DO
```

ويكمل البرنامج كالمعتاد.

2-2 نظرية الهيكلة

نظرية الهيكلة تشكل أساسا لمنهج البرمجة المهيكلة، وتنص على أنه يمكن صياغة البرامج الحاسوبية باستخدام ثلاثة هياكل أساسية حاكمة فقط، تمثل بكل سهولة في الخوارزم: المتتابعات، والاختيار، والتكرار.

1- الأوامر المتتابعة

هيكل التتابع يضمن التنفيذ التتابعي للأوامر أمرا وراء الآخر، وتصاغ الأوامر المتتابعة في أسطر متتالية، على الشكل التالي:

statement1
statement2
statement3

وهكذا.

ويمكن أن تستخدم الأوامر المتتابعة أيا من الأربع عمليات الأول المبينة آنفا، فقد يأخذ جزء من خوارزم الشكل التالي:

Add 1 to page_count
Print heading line
Set line_count to zero
Read customer record

2- الاختيار

هيكل الاختيار يحقق أن يتم الاختيار بين إجراءين، بحسب تحقق شرط ما أو عدم تحققه. ويمثل هذا الهيكل إمكانية اتخاذ القرارات في البرمجة، ويستخدم العملية الأساسية الخامسة المذكورة آنفا. وهذا الهيكل يأخذ الصورة التالية:

```
IF condition p (is true) THEN
        statement(s) in true case
Else
        statement(s) in false case
ENDIF
```

ومن الممكن أن يأخذ هذا الهيكل شكلا مختصرا، وذلك بحذف الجزء الخاص بالخيار الثاني، فيكون على الصورة التالية:

```
IF condition p (is true) THEN
        statement(s) in true case
ENDIF
```

وفي هذه الحالة فإنه إذا لم يكن الشرط متحققا، ينتقل التنفيذ للأمر التالي لهيكل الاختيار مباشرة.

3- التكرار

يمكن تعريف هيكل التكرار بأنه تمثيل لمجموعة من العبارات (يطلق عليها "بلوك **block**") تنفذ بصورة تكرارية طالما كان شرط ما متحققا. هذا الهيكل يمثل العملية الأساسية السادسة، وصورته العامة هي:

```
DOWHILE condition p (is true)
          statement block
ENDDO
```

وتعتبر الدوارة loop من نوع DOWHILE دوارة متقدمة الاختبار leading decision loop، بمعنى أن اختبار الشرط يتقدم تنفيذ أية عبارة من عبارات البلوك، ويعمل المحدد delimiter النهائي، وهو الكلمة الركيزية ENDDO على استثارة اختبار الشرط مرة أخرى، فإن كان لا يزال متحققا تكرر البلوك، وهكذا دواليك إلى أن نصل لحالة عدم تحقق الشرط، فيتحول التحكم في تنفيذ البرنامج إلى العبارة التالية للمحدد النهائي.

ويمكن للدوارة المبينة أن تصاغ على الصورة التالية:

```
WHILE condition p (is true) DO
          statement block
ENDDO
```

وتؤدي لنفس النتيجة

ولنأخذ المثال السابق ذكره:

```
Set student_total to zero
DOWHILE student_total < 50
     Read student record
     Print student name, address to report
     Add 1 to student_total
ENDDO
```

فنلاحظ أنه يثير النقاط التالية:

- المتغير student_total قد استهل بالقيمة "صفر" قبل الدخول في الدوارة.
- طالما كانت قيمة المتغير المذكور أقل من 50 (الشرط متحقق) يكرر البلوك.

- تتزايد قيمة المتغير المذكور طالما تكرر البلوك.
- بعد خمسين تكرارا، ووصول قيمة المتغير إلى 50، وهي حالة عدم تحقق شرط الدوارة، توقف التكرار وخرجنا منها.

ومن المهم ملاحظة أن الاستهلال والتزايد المستمر طوال عملية التكرار هما خاصتان أساسيتان في هيكل الدوارة DOWHILE.

ملخص الفصل

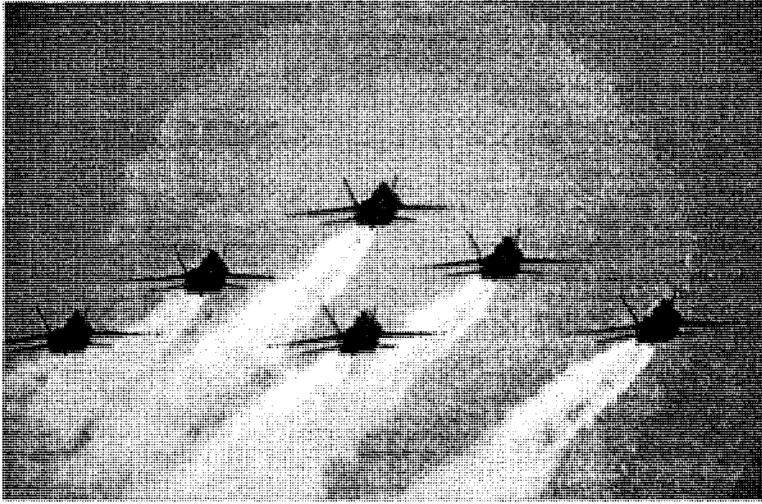
قدمنا في هذا الفصل ست عمليات أساسية يقوم بها الحاسوب، والكلمات الحاكمة والعبارات التي تمثل بها في اللغة شبه البرمجية. هذه العمليات هي: الإدخال، الإخراج، إجراء العمليات الحسابية، تخصيص القيم للبيانات، العبارات الشرطية والدوارات التكرارية. وقد أعطيت أمثلة واقعية لتلك العمليات.

ثم قدمنا نظرية الهيكلة، والتي تنص على أنه من الممكن صياغة أي برنامج حاسوبي باستخدام ثلاثة هياكل تحكم فقط، المتتابعات والاختيار والتكرار. وقد عرفنا كل هيكل منها في صيغته العامة، وربطناها بالعمليات الستة الأساسية للحاسوب.

---

[i] نسير في هذا الكتاب على ترجمة كلمة **increase** "*زيادة*" وكلمة **increment**، وهي الزيادة التدريجية بمقدار الوحدة "*تزايد*" -المترجم

الفصل الثالث وضع الخوارزم



الأهداف:

- توضيح طرق تحليل المسألة البرمجية ووضع الحل.
- وضع خوارزم بسيط باستخدام هيكل الأوامر المتتابعة.
- توضيح طريقة التحقق اليدوي من صحة الحل.

الخطوط العامة :

3-1 تحليل المسألة

قدمنا في الفصل الأول لسبع خطوات لوضع برنامج حاسوبي، وكانت الخطوة الأولى هي تحليل المسألة، وهي الخطوة الأكثر أهمية. ويتضمن ذلك القراءة المتأنية وتكرار القراءة للمسألة إلى أن يتم الفهم الكامل لما هو مطلوب. وفي كثير من الأحايين يتطلب الأمر معلومات خارجية لكشف غموض أو قصور في عرض المسألة.

ولمساعدتك على القيام بهذه الخطوة، قسم المسألة إلى ثلاثة أقسام:

1. المدخلات: قائمة بمصادر البيانات المتاحة للمسألة.
2. المخرجات: قائمة بالمخرجات المطلوبة.
3. المعالجة: قائمة بالأعمال المطلوبة لإنتاج المخرجات المطلوبة.

وخلال قراءة عبارات وصف المسألة، يمكن بسهولة تمييز المدخلات والمخرجات، حيث إنها كلمات أسماء وصفات. كما يمكن تمييز المعالجة بسهولة أيضا، فعبارات المسألة تصف خطواتها عادة على صورة نشاطات، مستخدمة الأفعال والظروف.

فلكي تقسم المسألة إلى الأقسام المذكورة، فإنك في الواقع تحلل عبارات وصف المسألة لغويا إلى ما يتضمن أسماء وصفات وإلى ما يتضمن منها أفعالا وأنشطة. وقد يساعدك في ذلك أن تقوم بوضع خطوط تميز بها الصفات، والأسماء، والأفعال، والظروف.

وفي بعض الأحايين يتم تحديد الأقسام الثلاثة تحديدا واضحا، وفي مثل هذه الحالات يكون من المستحسن التركيز على المخرجات المطلوبة، ويقودنا ذلك إلى أغلب المدخلات، ويؤدي ذلك بنا إلى تحديد العمليات المطلوبة لإنتاج المخرجات من المدخلات.

وعند هذه المرحلة يجب وضع قائمة بالعمليات المطلوبة، دون الدخول في تفاصيل كيفية القيام بها. لا تحاول وضع الحل قبل أن يتم تحديد المسألة تماما، ولنضرب مثالا بسيطا:

مثال 3-1 جمع ثلاثة أعداد

يراد وضع برنامج يقوم بقراءة ثلاثة أعداد وجمعهم وإخراج المجموع. ستجد أن صياغة المسألة على الوجه التالي:

A program is required to read three numbers, add them together, and print the total.

عالج هذه المسألة على مرحلتين، أولا؛ ضع خطا تحت الأسماء والصفات، لتمييز المدخلات والمخرجات، على الوجه التالي:

A program is required to read <u>three numbers</u>, add them together, and print the <u>total</u>.

ومن هذه الخطوة تعلم أن المدخلات هي ثلاثة أعداد، والمخرجات هي المجموع. ومن المفيد أن تضع ما توصلت إليه في صورة شكل يطلق عليه الجدول التحليلي، كالتالي:

Input	Processing	Output
number_1		total
number_2		
number_3		

الخطوة التالية هي أن تخطط تحت الأفعال والظروف (بلون مختلف أو بخط أكثر سمكا، أو بخطين كما نفعل في المثال)، فتجد أن الصياغة أخذت الشكل التالي:

A program is required to <u>read</u> <u>three numbers</u>, <u>add</u> them together, and <u>print</u> the <u>total</u>.

وبمراجعة الكلمات المخططة في هذه الخطوة، تجد أن الأفعال المبينة لعمليات المعالجة هي read, add, print. يمكن إذن إضافة هذه الخطوة إلى الجدول التحليلي، على الوجه التالي:

Input	Processing	Output
number_1	read three numbers	total
number_2	add numbers together	
number_3	print total number	

استخدام مسميات ذات معنى

يجدر التنويه هنا لأهمية استخدام المسميات المعبرة عن المضمون. فمسميات المتغيرات يستحسن أن تكون معبرة عن ماهيتها بقدر الإمكان، أخذا في الاعتبار أنها مجرد إشارات لأماكن في الذاكرة مخصصة لحفظ قيم ذلك المتغير. فتسمية مثل number_1, number_2, number_3 أفضل من إعطاء المتغيرات تسميات من قبيل A,B,C.

وحينما نأتي إلى تسميات العمليات، يجب عليك استخدام كلمات تصف العمل المطلوب تنفيذه بوضوح وتحديد. ففي مثالنا المعطى تلاحظ أننا ألحقنا فعل الأمر بمفعوله في جمل بسيطة. وقد بينت الدراسات أن اتخاذ هذا المنهج يحقق ميزتين هامتين؛ أولا، وضع منهجية سلسة لتحديد المسألة، وثانيا، تحليل العملية إلى عناصر واضحة. وهذه الميزة الأخيرة سوف يتضح خطرها عند الحديث عن الوحدات البنائية.

مثال 3-2 إيجاد متوسط درجات حرارة

لنأخذ مثال برنامج يسأل المستخدم أن يدخل درجات الحرارة العليا والدنيا، ثم يوجد متوسطها، مصاغ على الوجه التالي، (بعد أن أجريت علي صياغته عملية التحليل):

A program is required to prompt the terminal operator for the maximum and minimum temperature readings on a particular day, accept those readings as integers, and calculate and display to the screen the simple average temperature, calculated by (maximum + minimum)/2.

ومن التحليل يكون جدول التحليل على الوجه التالي:

Input	Processing	Output
max_temp	Prompt for temperature	avg_temp
	Get max, min temperature	
	Calculate average temperature	
	Display average temperature	

ولتتذكر أنه في هذه المرحلة لا نكون مهتمين بطريقة حساب المتوسط، فذلك سوف يؤخذ في الاعتبار عند وضع الخوارزم.

مثال 3-3 حساب وقت قص الحشائش

مطلوب برنامج يحسب المدة الزمنية اللازمة لقص حشائش حديقة منزل، باستخدام بيانات عن أبعاد المساحة الكلية، وأبعاد مساحة المباني، ووقت قص المتر المربع. وإليك صياغة البرنامج مع التحليل:

A program is required to read the length and width of the block, and the length and width of the house which is built on the block. The algorithm should compute and display the time required to cut the grass around the house, at a rate of 2 square meters per minute.

Input	Processing	Output
block_length	Prompt for measurements	mowing_time
block_width	Get block measurements	
house_length	Calculatehouse measurements	
house_width	Calculate mowing time	
	Display mowing time	

والخطوات المبينة كافية لحساب الوقت المطلوب. ويجب عليك أن تكون واثقا تمام الثقة من خطوات برنامجك قبل أن تحاول التفكير في طريقة تنفيذ هذه الخطوات.

3-2 تصميم خوارزم الحل

إن تصميم خوارزم الحل هو أكثر مراحل البرمجة إثارة للتحدي. فبعد أن تكون قد انتهيت من تحليل المسألة بوضوح، يمكنك أن تبدأ في إجراء تصور لطريقة الحل. انظر للمطلوب، ومنه وعن طريق الهياكل الثلاثة التي تنص عليها نظرية الهيكلة، حاول أن تضع خطة لخطوات المعالجة.

ولا تكون الخطوة الأولى في وضع الخوارزم هي النهائية في أغلب الأحوال، فكثيرا ما يتطلب الأمر إرجاء بعض الخطوات، وتعديل البعض الآخر، أو محوها. وصياغة الخوارزم باللغة شبه البرمجية مفيد في مرحلة التجربة والخطأ هذه، فمن السهل إجراء التعديلات عليه. فلا تتردد في

إجراء أية تعديلات تراها ضرورية في الخوارزم، حتى ولو تلغيه كلية وتبدأ التفكير من جديد إذا لم تكن راضيا عنه تمام الرضا. فإذا لم يكن الخوارزم سليما، فلن يكون هناك برنامج على الإطلاق.

وقد ظهر رأي يقول إنه بوضع الخوارزم ينتهي عمل المبرمج، حيث يمكن أن تترك عملية صياغة البرنامج الفعلي باللغة المطلوبة للمتدربين أو للمبتدئين، ولا يحدث ذلك عمليا، ولكن من المهم ألا تنشط لصياغة البرنامج قبل أن تتحقق تماما من سلامة الخوارزم الذي يوضح منطقه.

ونقدم لك فيما يلي الخوارزمات الخاصة بالأمثلة التي تم إعطاؤها، مستنبطة من الجدول التحليلي للمسألة defining diagram.

مثال 3-4 خوارزم جمع الأعداد

A program is required to read three numbers, add them together, and print the total.

A- defining diagram

Input	Processing	Output
number_1	read three numbers	total
number_2	add numbers together	
number_3	print total number	

يبين هذا الشكل ما هو مطلوب، وأن عملية رياضية بسيطة هو ما يحقق ذلك، وبصياغة الخوارزم باستخدام هيكل التتابع، نبين كيف يمكن تحقيقه.

B- Solution algorithm

```
Add_three_numbers
        Read number_1, number_2, number_3
        total = number_1+ number_2+ number_3
        Print total
END
```

ونلاحظ على هذا الخوارزم البسيط ما يلي:

- أن الخوارزم قد أعطي اسما هو Add_three_numbers، يصف باختصار الهدف منه. والاسم يتكون عادة من فعل يتبعه اسم أو اسمان.

- استخدمت الكلمة END لتبين انتهاء الخوارزم.
- استخدم أسلوب الإزاحة لكافة العبارات الخاصة بالخوارزم ما بين الاسم وكلمة الانتهاء، تسهيلا للقراءة.
- كل خطوة في المعالجة في الجدول التحليلي ترتبط مباشرة بعبارة من عبارات الخوارزم أو أكثر. فمثلاً، السطر read three numbers قد ترجم في الخوارزم إلى Read number_1, number_2, number_3، والسطر add numbers together قد ترجم إلى total = number_1+ number_2+ number_3.

وبعد أن انتهينا من وضع الخوارزم، علينا أن نبدأ في اختباره مكتبيا، ولكننا سوف نعالج ذلك في القسم التالي.

مثال 3-5 خوارزم متوسط الحرارة

A program is required to prompt the terminal operator for the maximum and minimum temperature readings on a particular day, accept those readings as integers, and calculate and display to the screen the simple average temperature, calculated by (maximum + minimum)/2.

A- defining diagram

Input	Processing	Output
max_temp	Prompt for temperature	avg_temp
	Get max, min temperature	
	Calculate average temperature	
	Display average temperature	

ونستخدم في الخوارزم لهذه المسألة الهيكل التتابعي، وعملية حسابية بسيطة، على الوجه التالي:

B- Solution algorithm

```
Find_average_temperature
        Prompt operator for max_temp, min_temp
        Get max_temp, min_temp
        avg_temp = (max_temp +min_temp).2
        Output avg_temp to the screen
```

END

وكما ترى، فإن العبارة Calculate average temperature في الجدول التحليلـــــي قـــد صيغت في الخوارزم كعملية حسابية لحساب المتوسط.

مثال 3-6 خوارزم حساب وقت قطع الحشائش

A program is required to read the length and width of the block, and the length and width of the house which is built on the block. The algorithm should compute and display the time required to cut the grass around the house, at a rate of 2 square meters per minute.

A- defining diagram

Input	Processing	Output
block_length	Prompt for measurements	mowing_time
block_width	Get block measurements	
house_length	Calculate house measurements	
house_width	Calculate mowing time	
	Display mowing time	

أيضا في هذا الخوارزم سوف يستخدم الخوارزم هيكل التتابع، والذي يبـــين خطـــوات تنفيــذ الأوامر واحدا بعد الآخر، وسوف تعطى أسماء ذات معنى للمتغيرات المعبرة عن بيانات المســـألة المعطاة.

B- Solution algorithm

```
Calculate_mowing_time
        Prompt user for block_length, block_width
        Get block_length, block_width
        block_area = block_length* block_width
        Prompt user for house_length, house_width
        Get house_length, house_width
        house_area = house_length* house_width
        mowing_area = block_area - house_area
        mowing_time = mowing_area/2
        Output mowing_time to screen
END
```

3-3 اختبار الخوارزم

بعد وضع الخوارزم من المفروض أن يجتاز اختبار الصلاحية، وهي خطوة هامة حيث إن أغلب الأخطاء المنطقية الجسيمة تحدث في مرحلة وضع الخوارزم، وإذا لم تكتشف في حينها فسوف تعرقل تنفيذ البرنامج. ومن الأيسر بكثير أن تكتشف في الخوارزم عن أن تكتشف في البرنامج الواقعي وقت التنفيذ، ذلك لأنه ما أن يبدأ تشغيل البرنامج حتى يفترض سلامة منطقه، مما يدفع بمحاولات كشف الأخطاء في اتجاهات تكون بعيدة عن مصدر الخلل، ومن جهة أخرى، من الصعب الرجوع لهيكل منطق البرنامج بعد صياغته نهائيا، فتبذل ساعات طوال من الإحباط في هذا المجهود.

ويعني الاختبار المكتبي للخوارزم تتبع تنفيذ خطواته عن طريق تطبيقها على قيم اختبارية يضعها مصمم الخوارزم، بمعنى تخيل ما يقوم به الحاسوب عمليا، ومحاولة تمثيله في ذلك، مع تتبع التغير في قيم المتغيرات خلال هذا التمثيل. ولا يفيد ذلك فقط في اكتشاف الأخطاء، بل يزيد المبرمج فهما لكيفية تشغيل البرنامج على الحاسوب. فكلما ازددت قربا من تصور التنفيذ، كان أيسر لك اكتشاف الأخطاء.

اختيار القيم لإجراء الاختبار

حينما تختار قيما لاختبار الخوارزم، عليك الرجوع لمنطوق مسألة البرنامج وتخير قيما بسيطة بناء ذلك المنطوق، وليس على الخوارزم نفسه. وبذلك تستمر في تركيزك على متطلبات البرنامج، وليس على كيفية التنفيذ.

ولإجراء الاختبار المكتبي، عليك أن تختار فقط بعض الحالات البسيطة التي تبين لك الخطوط العريضة لسير الخوارزم. فالاختبار النهائي والشامل للبرنامج سوف يتم بعد صياغته في لغته البرمجية. وتتلخص خطوات الاختبار المكتبي فيما يلي:

1. اختر قيما بسيطة كمدخلات، ويكفي عادة حالتين اختباريتين أو ثلاث.

2. ضع الناتج المتوقع من البرنامج لكل حالة، وهنا سيفيدك اختيار مدخلات بسيطة، فمن السهل جمع 10 و20 و30 عن جمع 2.5 و13.4 و21.1.
3. اصنع جدولا مبينا به المتغيرات الموجودة في الخوارزم.
4. تتبع قيم لمتغيرات خلال تطبيق خطوات الخوارزم عليها، وسجل هذه القيم خطوة بعد الأخرى.
5. كرر نفس الخطوات مع الحالة الثانية
6. في كل حالة قارن الناتج المتوقع في الخطوة 2 مع النتائج التي تحصل عليها.

وسوف يكشف لك الاختبار المكتبي عن أغلب الأخطاء، ولكن ليس دائما كلها.

والآن، لنطبق المبادئ السابقة على الأمثلة المعطاة.

مثال 3-7 اختبار خوارزم جمع الأعداد

لنستعد خوارزم الحل:

A- Solution algorithm

```
Add_three_numbers
        Read number_1, number_2, number_3
        total = number_1+ number_2+ number_3
        Print total
END
```

ولإجراء الاختبار نتبع الخطوات التالية:

B- desk checking

1) نختار مجموعتين من الأعداد كمدخلات تجريبية، نراعي فيها البساطة:

Input data

	First data set	Second data set
number_1	10	40
number_2	20	41
number_3	30	41

2) نضع تصورا للنتيجة لكل مجموعة:

Expected results

	First data set	Second data set
total	60	123

3) نصنع جدولا يضم أسماء المتغيرات، وخطوات الخوارزم، ونطبقها على كل مجموعة خطـــوة بعد الأخرى على النحو التالي:

Desk check table

Statement		number_1	number_2	number_3	total	print
First pass	**Read**	**10**	**20**	**30**		
	total				**60**	
	Print					**yes**
Second pass	**Read**	**40**	**41**	**42**		
	total				**123**	
	Print					**yes**

4) نقارن النتائج المتحققة بالنتائج المتوقعة.

وبعد إجراء هذا الاختبار، يمكن البدء في صياغة البرنامج باللغة المختارة.

وبنفس الطريقة سوف نجري الاختبار المكتبي على الخوارزمات الخاصة ببقية الأمثلة، وهي علـــى بساطتها، لكونها تعتمد على هياكل تتابعية، تعطي فكرة عن مفهوم هذا الاختبار.

مثال 3-8 اختبار خوارزم متوسط درجة الحرارة

A- Solution algorithm

```
Find_average_temperature
        Prompt operator for max_temp, min_temp
        Get max_temp, min_temp
        avg_temp = (max_temp +min_temp).2
        Output avg_temp to the screen
END
```

B- Desk checking

1) اختيار المدخلات التجريبية

Input data

	First data set	Second data set
max_temp	30	40
min_temp	10	20

2) النتائج المتوقعة

Expected results

	First data set	Second data set
avg_temp	20	30

3) الاختبار المكتبي

Desk check table

Statement		prompt	max_temp	min_temp	avg_temp	output
First pass	Prompt	yes				
	Get		30	10		
	Calculate				20	
	Output					yes
Second pass	Prompt	yes				
	Get		40	20		
	Calculate				30	
	Output					yes

4) قارن النتائج المتوقعة بالنتائج التي يحصل عليها من الاختبار.

مثال 3-9 اختبار خوارزم حساب وقت قص الحشائش

A- Solution algorithm

```
Calculate_mowing_time
        Prompt user for block_length, block_width
        Get block_length, block_width
        block_area = block_length* block_width
        Prompt user for house_length, house_width
        Get house_length, house_width
        house_area = house_length* house_width
        mowing_area = block_area - house_area
        mowing_time = mowing_area/2
        Output mowing_time to screen
END
```

B- Desk checking

1) اختيار المدخلات التجريبية

Input data

	First data set	Second data set
block_length	30	40
block _width	30	20
house_length	20	20
house_width	20	10

2) النتائج المتوقعة

Expected results

	First data set	Second data set
mowing_time	250 minutes	300 minutes

3) الاختبار المكتبي

Statement	blockl ength	block width	house length	house width	block area	house area	mowing area	mowing time
First pass								
Get	30	30						
block_area					900			
Get			20	20				
house_area						400		
mownig_area							500	
mowing_time								250
Output								yes
Second pass								
Get	40	20						
block_area					800			
Get			20	10				
house_area						200		
mowing_area							600	
mowing_time								300
Output								yes

4) تقارن النتائج المتوقعة بالنتائج التي يحصل عليها من الاختبار.

3-4 ملخص الفصل

تناولنا في الجزء الأول من الفصل طريقة تحليل المسألة البرمجية وتحديدها، فيجب عليك أن تحيط بأية مسألة إحاطة تامة قبل أن تفكر في وضع حل لها. والطريقة المقترحة تتلخص في تحليل

الكلمات الفعلية لتوصيف المسألة، بهدف تحليل المسألة إلى ثلاثة عناصر: المدخلات، المخرجات، المعالجة. وقد أعطينا عدة أمثلة توضح كيفية الحصول على جدول تحليل المسألة، وقد شددنا على أن خطوات المعالجة يجب أن تسجل ما يراد تنفيذه وليس كيفية هذا التنفيذ.

وفي الجزء الثاني من الفصل ركزنا على كيفية وضع خوارزم الحل. فبعد إجراء التحليل للمسألة المعروضة يبدأ المرء في التفكير في محاولة إيجاد الحل معبرا عنه بخوارزم، ولعمل ذلك يجب استخدام عبارات صحيحة للخوارزم. وقد اكتفينا في هذا الفصل بأمثلة تستخدم الهيكل التتابعي في التحكم في البرنامج.

وقد خصص الجزء الثالث لعملية الاختبار المكتبي للخوارزم، وذلك بتمثيل دور الحاسوب في تنفيذه باستخدام بيانات تجريبية، وطبق ذلك على الأمثلة المعطاة.

تدريبات

في المسائل الآتية مطلوب منك:

1. تحليل المسألة عن طريق جدول التحليل
2. وضع الخوارزم بأسلوب اللغة شبه البرمجية
3. إجراء الاختبار المكتبي لصحة الخوارزم

1- ضع خوارزما يستحث المستخدم على إدخال ثلاثة أحرف، ويتقبلها كمدخلات، ثم يظهر رسالة ترحيب على الصورة "مرحبا س س س"

1- Construct an algorithm which will prompt the user to input three characters, receive those characters, and display a welcoming message such as: "Hello xxx".

2- يراد وضع برنامج يتقبل عددين صحيحين من المستخدم، ويظهر مجموعهما وحاصل طرحهما وحاصل ضربهما وخارج قسمتهما.

A program is required which will receive two integer items from a terminal operator and display to the screen their sum, deference, product and quotient.

3- يراد وضع برنامج يقرأ نسبة الضريبة وأثمان خمسة أصناف، ويحسب مجموع الثمن قبل الضريبة ومقدار الضريبة المستحقة محسوبة عن طريق ضرب نسبة الضريبة في مجموع الثمن، وتطبع القيمتان.

A program is required which will read a tax rate (as a percentage) and the prices of five items. The program is to calculate the total price, before tax and them the tax payable computed by applying the tax rate to the total price. Both values are to be printed as output.

4- يراد وضع برنامج يقرأ رصيد عميل أول الشهر، ومجموع مسحوباته خلال الشهر، ومجموع إيداعاته خلال الشهر. وتوجد ضريبة على كافة المعاملات مقدارها 1%، والمطلوب أن يحسب البرنامج الرصيد في نهاية الشهر عن طريق إجراء العمليات التالية على رصيد أول الشهر: 1) طرح مجموع المسحوبات 2) إضافة مجموع الإيداعات، 3) طرح الضريبة ، ويطبع البرنامج الرصيد آخر الشهر.

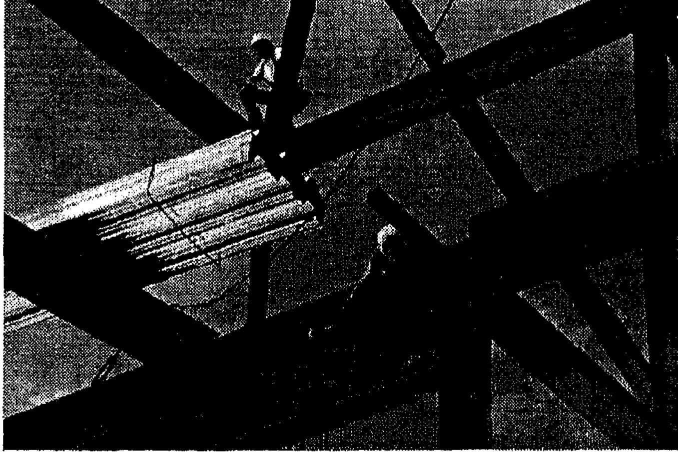
A program is required which will read in one customer's account at the beginning of the month, a total of all withdrawals, and a total of all deposits made during the month. A tax charge of 1% is to be applied is all transactions made during the month. The program is to calculate the account balance at the end of the month by 1) subtracting the total withdrawals, 2) adding the total deposits 3) subtracting tax form the account balance at the beginning of the month. after calculation, the balance at the end of the month is to be printed.

5- يراد وضع برنامج يقرأ ساعات عمل عامل في أسبوع، ويحسب ويطبع أجره المستحق عن ذلك الأسبوع. القيم المقروءة هي عدد ساعات العمل العادية، وعدد الساعات الإضافية، والأجر في الساعة. يحسب الأجر عن الساعات الإضافية بواقع مرة ونصف مقدار الأجر العادي.

A program is required which will read in the values from an employee's time sheet, and calculate and print the weekly pay owing to that employee. The values read in are the total number of regular

hours, the total overtime hours, and the hourly wage rate. Payment for overtime hours is to be computed at time-one-a half.

الفصل الرابع هيكل الاختيار



الأهداف:

- التدرج في هياكل خوارزمات الاختيار من الاختيار البسيط للمركب، فالمتداخل
- عرض حالة الاختيار بين أكثر من بديلين (العبارة CASE)
- وضع خوارزمات تحتوي على مزيج من هياكل الاختيار المختلفة

الخطوط العامة :

4-1 هيكل الاختيار

4-2 الخوارزمات المحتوية على هيكل اختيار

4-3 هيكل الاختيار بين أكثر من بديلين

4-4 ملخص الفصل

4-5 تدريبات

1-4 هيكل الاختيار

قدمنا لهيكل الاختيار في الفصل الثاني، في معرض الحديث عن نظرية الهيكلة، وهذا الهيكل يمثل قدرة الحاسوب على اتخاذ القرارات، بمعنى أنه يمكنك استخدام هذا الهيكل للتحكم في تنفيذ البرنامج بحيث يتخذ أحد مسارين أو أكثر، بحسب تحقق شرط ما من عدمه. وهناك أكثر من صورة لهذه الإمكانية.

1- الاختيار البسيط

يكون الاختيار بسيطا حينما يحدد له أحد بديلين، ويستخدم للتعبير عن ذلك في الخوارزم الكلمات F, ELSE, THEN كما عرضنا في الفصل الثاني، وكمثال على ذلك العبارات التالية:

```
IF account_balance < $300      HTEN
        service_charge = $5.0
ELSE
        service_charge = $2.0
ENDIF
```

2- الاختيار البسيط غير المتفرع

يقصد بذلك عبارة الاختيار التي لا تحتوي على بديل ثان، وبالتالي لا تستخدم كلمة ELSE. فإذا لم يكن الشرط متحققا، انتقل التنفيذ إلى الخطوة التالية لهيكل الاختيار مباشرة، ومن أمثلة ذلك:

```
IF student_attendance = part_time THEN
        add 1 to part_time_count
ENDIF
```

وفي هذه الحالة سيتم تزايد المتغير part_time_count فقط إذا تحقق الشرط المذكور، بأن كان المتغير student_attendance تساوي قيمته part_time. وفي غير هذه الحالة يتخطى هيكل الاختيار ويسير البرنامج في خطة كما لو كان الهيكل غير موجود.

3- الاختيار المركب

يقصد بهيكل الاختيار المركب combined (compund) عدة شروط مترابطة بعلاقات منطقية من نوع الجمع "و AND" أو التخيير "أو OR". وفي الحالة الأولى يجب تحقق الشرطين حتى يؤخذ قرار معين، وفي الحالة الثانية يكفي تحقق أي من الشرطين لاتخاذ القرار. ومن أمثلة الحالة الأولى:

```
IF student_attendance = part_time
AND student_geder = female THEN
        add 1 to fem_part_time
ENDIF
```

وفي هذه الحالة سوف يجرى على سجل الطلبة نوعين من الاختبار، طبقا للشرطين المذكورين في البرنامج، بحيث لا تتزايد قيمة المتغير fem_part_time [المعبر عن طالبة غير منتظمة] إلا بتحقق الشرط الأول، وهو: student_attendance = part_time [حضور الطالب = غير منتظم] و الشرط الثاني، وهو:student_geder = female [جنس الطالب = أنثي].

ولو أننا غيرنا العلاقة المنطقية لتكون OR بدلا من AND على الوجه التالي:

```
IF student_attendance = part_time
OR student_geder = female THEN
        add 1 to fem_part_time
ENDIF
```

لوجدنا أن المتغير fem_part_time سوف يتزايد في حالة تحقق أي من الشرطين المذكورين، أي في حالة كون الطالب غير منتظم، أو جنسه أنثى، بمعنى أن الشرط المركب ينطبق على الحالات التي يكون الطالب فيها إما أنثى منتظمة أو أنثى غير منتظمة، أو ذكر غير منتظم. في هذه الحالة يكون اختيار اسم المتغير خاطئا، لأنه يشمل حالات لا تتفق مع التسمية. والهدف من هذا التحليل التركيز على أهمية فهم الشروط المركبة فهما جيدا لإمكان التعامل معها تعاملا صحيحا.

ويمكن للشروط المركبة أن تضم علاقات مختلطة بين الجمع والتخيير. على أنه إذا ما اجتمع النوعان في نفس عبارة الشرط IF، فلا بد من استخدام الأقواس منعا للبس، وإليك المثال التالي:

```
IF record_code = '23'
OR update_code = delete
AND account_balance = zero THEN
        delete customer record
ENDIF
```

بالنظر إلى الشروط السابقة، والتي تحدد حالة محو سجل عميل، نجد أنها تثير اللبس في تطبيقها، فليس معلوما كيف يطبق التخيير، هل يؤخذ مع الشرطين الأول والثاني كمجموعة، ثم تجرى عملية الجمع مع ناتج عملية التخيير، أم بين الشرط الأول من جهة والشرطين الثاني والثالث كمجموعة، [خذ حالة تحقق الشرط الأول والثاني وعدم تحقق الشرط الثالث، لو اعتبرت أن التخيير هو بين الشرطين الأول والثاني كمجموعة مضافا إليها الشرط الثالث، فلن يحذف الملف لعدم تحقق الجمع مع أي منهما، بينما لو اعتبرت أن التخيير بين الشرط الأول من جهة، والشرطين الثاني والثالث مجموعين معا من جهة أخرى، فسوف يحذف طبقا لتحقق التخيير].

ويعالج هذا الموقف باستخدام الأقواس، فالحالة الأولى يعبر عنها بالشكل التالي:

```
IF (record_code = '23'
OR update_code = delete)
AND account_balance = zero THEN
        delete customer record
ENDIF
```

بينما يعبر عن الحالة الثانية بالصورة التالية:

```
IF record_code = '23'
OR (update_code = delete
AND account_balance = zero) THEN
        delete customer record
ENDIF
```

4- الشروط المتداخلة

تحدث الشروط المتداخلة nested عندما تتوالى، بحيث تحتوي عبارة الشرط على شرط آخر، وتكون على صورتين:

التداخل الخطي

تسمى هذه الصورة بالتداخل الخطي linear لأن كل ELSE تكون تالية لشرطها مباشـــرة، على الشكل التالي:

```
IF record_code = 'A' THEN
        increment counter_A
Else
        IF record_code = 'B' THEN
                increment counter_B
        ELSE
                IF record_code = 'C' THEN
                        increment counter_C
                ELSE
                        increment error_counter
                ENDIF
        ENDIF
ENDIF
```

وبمتابعة منطق الخوارزم نجد أن كل الشروط يتم تقييمها وتنفيذ القرار الناتج عنها بالتتابع، بمعنى أن الاختيار يتم في أية لحظة يتحقق فيها أي من الشروط. فإذا ما كان كود السجل من النـــوع A، يتم تزايد المتغير الخاص به، وإلا ينتقل الاختبار للشرط التالي، أن يكون السجل من النـــوع B، ثم النوع C، ثم إلى المتغير المعبر عن الخطأ، في حين كون السجل المختبر لا ينتمي لأي مـــن الأنواع الثلاثة.

التداخل غير الخطي

في التداخل غير الخطي nonlinear يجرى تقييم عدة شروط بالتتابع أولا، ثم تحدد الخيـــارات المتخذة، وذلك كما في المثال التالي:

```
IF student_attemdance = part_time THEN
        IF student_gender = female THEN
                IF student_age > 21 THEN
                        add 1 to mature_fem_pt_time
                Else
                        add 1 to young_fem_pt_time
                ENDIF
```

```
        ELSE
                add 1 to male_pt_time
        ENDIF
ELSE
        add 1 to ful_time
```

الهدف من هذه القطعة من خوارزم برنامج هو تحديد طبيعة الطالب، من حيث كونه ذكرا أو أنثى، وفي الحالة الأخيرة من حيث كونها قاصرا أو بالغا، وفي الحالتين من حيث حالة الانتظام. وتلاحظ الاختلاف في هذه الصورة عن الصورة السابقة، حيث تتوالى الشروط أولا، ثم تتلوها الخيارات. وفي الصورتين يجب الانتباه جيدا إلى تساوي عدد الشروط (عبارة IF) مع الخيارات (عبارة ELSE). ويساعد الاستخدام الجيد للإزاحة في توضيح منطق البرنامج ومتابعته.

على أن التداخل غير الخطي من مناطق الزلل الشائعة، ومن المفروض أن يستخدم بحرص وفي أضيق الحدود. ومن الممكن في حالات كثيرة الاستغناء عنه باستخدام التداخل المركب، حيث إن عبارتين شرطيتين متتابعتين يمكن أن يصاغا كشرط مركب باستخدام الرابطة "و". والمثال التالي يبين ذلك:

لنفرض أن لدينا عبارة شرطية متداخلة تداخلا غير خطي على النحو التالي:

```
IF student_attendance = part_time THEN
        IF student_age > 21 THEN
                increment mature_pt_time
ENDIF
```

تجد أنها يمكن أن تصاغ كشرط مركب على الصورة التالية:

```
IF student_attendance = part_time
AND student_age > 21 THEN
                increment mature_pt_time
ENDIF
```

وتحصل على نفس النتيجة.

2-4 خوارزمات الاختيار

لنأخذ بعض الأمثلة التي تحتوي على هياكل اختبار. وفي كل مثال سوف تحلل المشكلة، ويوضع لها الخوارزم، ثم يختبر الخوارزم يدويا. وللتبسيط سوف نكتفي بتخطيط الأفعال المعـــبرة عـــن عمليات المعالجة.

مثال 4-1 ترتيب ثلاثة أحرف

المطلوب تصميم برنامج يتقبل ثلاثة أحرف ويرتبهم تصاعديا.

Design a program which will prompt the user for three characters, accept those characters as inputs, sort them into ascending sequence, and output them on the screen.

A- defining diagram

Input	Processing	Output
char_1	Prompt for characters	char_1
char_2	Accept three characters	char_2
char_3	Sort three characters	char_3
	Output three characters	

هذا ويتطلب خوارزم الحل سلسلة من العبارات الشرطية لكي يتم الترتيب، ونستخدم لذلـــك متغيرا يسمى temp تخزن فيه قيم الحروف مؤقتا أثناء عملية الترتيب، على الوجه الآتي:

B- Solution algorithm

```
Sort_three_characters
        Prompt the user for char_1, char_2, char_3
        Get char_1, char_2, char_3
        IF char_1 > char_2 THEN
                temp = char_1
                char_1 = char_2
                char_2 = temp
        ENDIF
        IF char_2 > char_3 THEN
                temp = char_2
                char_1 = char_3
```

```
            char_3= temp
        ENDIF
            IF char_1 > char_2 THEN
            temp = char_1
            char_1 = char_2
            char_2 = temp
        ENDIF
        Output char_1, char_2, char_3
END
```

تتم عملية الترتيب على الوجه التالي:

- يقارن المتغير char_1 (المحتوي على الحرف الأول) مع المتغير char_2 (المحتوي على الحرف الثاني)، فإذا كان الحرف الأول أكبر في الترتيب الأبجدي، يخزن في متغير التخزين المؤقت temp ويحل الثاني (الأصغر) محله في المتغير char_1، ثم يخزن المتغير الأكبر في المتغير char_2.
- بنفس الطريقة يجري استبدال المتغيرين char_2 وchar_3 إذا كان الترتيب بينهما منعكسا.
- يجرى نفس الاختبار الأول مرة أخرى، لتصحيح الترتيب بين المتغيرين الأول والثاني إذا كان الترتيب بينهما منعكسا.

[يلاحظ أن مقارنة الترتيب الأبجدي للحروف يتم حسب قيمها العددية في كود الأسكي]

C- Desk checking

i- Input data

	First data set	Second data set
char_1	k	z
char_2	b	s
char_3	g	a

ii- Expected results

	First data set	Second data set
char_1	b	a
char_2	g	s

	First data set	Second data set
char_3	k	z

iii- Desk check table

Statement		char_1	char_2	char_3	temp	IF executed?
First Pass	Get	k	b	g		
	IF	b	k		k	yes
	IF		g	k	k	yes
	IF					no
	Output	yes	yes	yes		
Second pass	Get	z	s	a		
	IF	s	z		z	yes
	IF		a	z	z	yes
	IF	a	s		s	yes
	Output	yes	yes	yes		

وتلاحظ أننا في جدول الاختبار أعطينا كل خطوة اختبار (IF) سطرا مستقلا يحدد إذا ما كـان الشرط متحققا، والوضع بعد هذه الخطوة بالنسبة لقيم المتغيرات الثلاثة، والمتغير المؤقت.

مثال 4-2 معالجة سجل عميل

يراد وضع برنامج يتلقى اسم العميل، ومقدار طلبيته، والكود الضريبي، ويحسب القيمة الإجمالية للفاتورة.

A program is required to read a customer's name, a purchase amount and a tax code which will be one of the following:

0 tax exempted
1 tax 3%
2 tax 5%
3 tax 7%

The program must then compute the sales tax and total amount due and print the customer's name, purchase amount, sales tax and total amount due.

A- Defining diagram

Input	Processing	Output
cust_name	Read customer details	cust_name
purch_amt	Compute sales tax	purch_amt

Input	Processing	Output
tax_code	Compute total amount	sales_tax
	Print customer details	total_amt

B- Solution algorithm

```
Process_customer_record
        Read cust_name, purch_amt, tax_code
        IF tax_code = 0 THEN
              sales_tax =0
        ElSE
              IF tax_code = 1 THEN
                   sales_tax = purch_amt*0.03
              ELSE
                   IF tax_code = 2 THEN
                        sales_tax = purch_amt*0.05
              ELSE
                        IF tax_code = 3 THEN
                          sales_tax = purch_amt*0.07
                        ENDIF
              ENDIF
        ENDIF
        total_amt = purch_amt + sales_tax
        Print customer's name, purchase amount, sales tax, total
amount
END
```

C- Desk checking

i- Input data

	First data set	Second data set
purch_amt	10.00	20.00
tax_code	0	2

ii- Expected results

	First data set	Second data set
sales_tax	0	1.0
total_amt	10.00	21.00

iii- Desk check table

Staement	purch_amt	tax_code	sales_tax	total_amt	IF?
First pass					
Read	10.00	0			
IF			0		IF tax_code =0
total_amt				10.00	
print	yes		yes	yes	
Second pass					
Read	20.00	2			
IF			1.00		IF tax_code =2
total_amt				21.00	
print	yes	yes		yes	

وتلاحظ أننا أدمجنا العبارات الشرطية (عدد 13 سطرا) في خطوة واحدة في جدول الاختبار.

مثال 4-3 حساب الأجر

تريد شركتك أن تضع لها برنامجا يتقبل رقم الموظف والأجر في الساعة وعدد ساعات العمل في الأسبوع، ويحسب أجر العامل، علما بأن الحد الأقصى لساعات عمل العامل هي 60 ساعة أسبوعيا، والحد الأقصى لأجر الساعة 25 جنيها، وعدد الساعات المعتادة هي 35 ساعة أسبوعيا. ويحسب الأجر الإضافي بواقع مرة ونصف الأجر المعتاد.

A program is required to <u>read</u> an employee's number, pay rate, and the number of hours worked in a week. The program is then to <u>compute</u> the employee's weekly pay and <u>print</u> it along with the input data.

According to the company's rules, no employee may work more than 60 hours per week, and the maximum hourly rate is $25.00 per hour. If more than 35 hours are worked, then payment for the overtime hours worked is calculated at time-and-a-half. If the hours worked field or the hourly rate is out of range, then the input data and an error message are printed without calculation made.

A- Defining diagram

Input	**Processing**	**Output**
emp_no	Read employee details	emp_no
pay_rate	Validate input fields	pay_rate
hrs_worked	Calculate employee pay	hrs_worked

Input	Processing	Output
	Print employee details	emp_pay
		error_mssage

وضع الخوارزم:

يتطلب خوارزم هذا البرنامج سلسلة من عبارات الشرط البسيطة والمتداخلة، وذلك للتحقق من الشروط المطلوبة، وهي ألا تزيد ساعات العمل عن 60 ساعة، والأجر اليومي عن 25، وهي الحدود القصوى التي وضعتها الشركة. فإذا تجاوزت القيمة المدخلة إحدى هاتين القيمتين، دل ذلك على وجود خطأ في الإدخال، وظهرت رسالة تحذيرية بذلك.

المتغيرات المنطقية:

يستلزم إجراء التحقق أن تضع متغيرا يسمى valid_input، وهو متغير من النوع *المنطقي* logic variable، ويطلق عليه أيضا "متغير بولي Boolean veriable"، بمعنى أنه يخضع لعلاقات الجبر البولي. والمتغير من هذا النوع يكون له إحدى قيمتين، صواب true وخطأ false.

وعلى ذلك تكون عبارة اختبار التحقق في مثالنا على الشكل التالي:

IF valid_input = true THEN

ويتلو ذلك الأوامر المنفذة في حالة تحقق الشرط. وقد تعارف المبرمجون على تيسير هذه الصيغة لتكون على الشكل التالي:

IF valid_input THEN

على أساس أن العبارة true = مفهومة ضمنا.

ويفهم من ذلك أن المتغيرات المنطقية تعمل في البرنامج عمل مفتاح للتحويل، فتحول مسار البرنامج من اتجاه لآخر بحسب حالة المتغير المنطقية.

وبناء على ما سبق يكون خوارزم الحل كما يلي:

B- Solution algorithm

```
Compute_employee_pay
        Set valid_input to true
        Set error_message to blank
```

```
    Read emp_no, pay_rate, hrs_worked
    IF pay_rate > 25.00 THEN
       error_message = "Pay rate exceeds $25.00"
       valid_input = false
       Print emp_no, pay_rate, hrs_worked, error_message
    ENDIF
    IF hrs_worked > 60 THEN
       error_message = "Hours worked exceeds limit of 60"
       valid_input = false
       Print emp_no, pay_rate, hrs_worked, error_message
    ENDIF
    IF valid_input THEN
       IF hrs_worked <=35 THEN
          emp_pay = pay_rate*hrs_worked
       ELSE
          overtime_hrs = hrs_worked - 35
          overtime_pay = overtime_hrs*pay_rate*1.5
          emp_pay = pay_rate*35 + overtime_pay
       ENDIF
       Print emp_no, pay_rate, hrs_worked, emp_pay
    ENDIF
END
```

وتلاحظ أن البرنامج في هذا الخوارزم ينقسم إلى قسمين متميزين، القسم الأول يبحث التحقق من عدم تجاوز الحدود الخاصة بساعات العمل والأجر الأسبوعي، والثاني هو الخاص بحساب الأجر، والذي لا ينتقل إليه البرنامج إلا في حالة كون نتيجة التحقق في الحالتين صوابا. وسنرى في الفصل السابع كيف يمكننا فصل كل قسم في وحدة بنائية خاصة.

C- Desk checking

i- Input data

	First data set	Second data set
pay_rate	10.00	40.00
hrs_worked	40	35

ii- Expected results

	First data set	Second data set
emp_pay	425	-
error_message	blank	Pay rate exceeds 25.00

iii- Desk check table

Statement	pay rate	hrs worked	overtime hrs	overtime pay	emp pay	valid input	error message	IF executed?
First Pass								
Initialize						true	blank	
Read	10	40						
IF								no
IF								no
IF			5	75	425			valid input
Print	yes	yes			yes			
Second pass								
Initialize						true	blank	
Read	40	35						
IF						false	pay rate exceeds $25	pay rate> 25
Print	yes	yes					yes	
IF								no
IF								no

3-4 هيكل العبارة CASE

هيكل العبارة CASE هو وسيلة أخرى للتعبير عن الاختيار بين عدة بدائل، تحل محل عبارات IF المتداخلة خطيا، ويستخدم هذا الهيكل لسببين: أنه أيسر في الفهم والتتبع، ومن الممكن تحويله مباشرة لكثير من اللغات الراقية.

والعبارة CASE ليست في الواقع هيكلا جديدا كلية، بل بالأحرى تبسيط لعملية الاختيار حينما يزيد عن احتمالين. ويأخذ الهيكل الخاص بها الصورة التالية:

```
CASE OF single variable
        value_1       : statement block_1
        value_2       : statement block_2



        value_n       : statement block_n
        value_other: statement block_other
```

```
ENDCASE
```

وللمقارنة بين الأسلوبين إليك هذا الجزء من البرنامج الذي سبق أن مر بك في دراسة التداخـــل الخطي:

```
IF record_code = 'A' THEN
        increment counter_A
Else
        IF record_code = 'B' THEN
                increment counter_B
        ELSE
                IF record_code = 'C' THEN
                        increment counter_C
                ELSE
                        increment error_counter
                ENDIF
        ENDIF
ENDIF
```

والآن سوف نصوغ نفس الجزء بأسلوب العبارة CASE:

```
CASE OF record_code
        "A"    : increment counter_A
        "B"    : increment counter_B
        "C"    : increment counter_C
        other  : increment error_counter
ENDCASE
```

وسوف نضع الآن هذا الأسلوب في مثال تطبيقي، وهو المثال الخاص بمعالجة سجل عميل.

مثال 4-4 معالجة سجل عميل

A program is required to <u>read</u> a customer's name, a purchase amount and a tax code which will be one of the following:
0 tax exempted
1 tax 3%
2 tax 5%
3 tax 7%

The program must then compute the sales tax and total amount due and print the customer's name, purchase amount, sales tax and total amount due.

A- Defining diagram

Input	Processing	Output
cust_name	Read customer details	cust_name
purch_amt	Compute sales tax	purch_amt
tax_code	Compute total amount	sales_tax
	Print customer details	total_amt

B- Solution algorithm

```
Process_customer_record
        Read cust_name, purch_amt, tax_code
        CASE OF tax_code
              0: sales_tax = 0
              1: sales_tax = purch_amt*0.03
              2: sales_tax = purch_amt*0.05
              3: sales_tax = purch_amt*0.07
        ENDCASE
        total_amt = purch_amt + sales_tax
        Print customer's name, purchase amount, sales tax, total
amount
END
```

C- Desk checking

i- Input data

	First data set	Second data set
purch_amt	10.00	20.00
tax_code	0	2

ii- Expected results

	First data set	Second data set
sales_tax	0	1.0
total_amt	10.00	21.00

iii- Desk check table

Staement	purch_amt	tax_code	sales_tax	total_amt	CASE?
First pass					
Read	10.00	0			
IF			0		CASE 0
total_amt				10.00	
print	yes		yes	yes	
Second pass					
Read	20.00	2			
IF			1.00		CASE 2
total_amt				21.00	
print	yes	yes		yes	

4-4 ملخص الفصل

غطينا في هذا الفصل هيكل الاختيار بالتفصيل. وقدمنا أمثلة له مع الخوارزمات الخاصة بها طبقنا بها الاختبار البسيط، وغير المتفرع، والمركب والمتداخل. وقد قدمنا عدة حلول بديلة لبعض المسائل.

كما قدمنا لعبارة CASE كهيكل بديل للاختيار المتداخل خطيا أكثر سهولة وإيجازا. والكثير من اللغات البرمجية تتضمن هذا الهيكل، ولذا فإن استخدامه في صياغة الخوارزمات يعتبر أمرا مستحبا.

5-4 تدريبات

في المسائل الآتية مطلوب منك:

- تحليل المسألة عن طريق جدول التحليل
- وضع الخوارزم بأسلوب اللغة شبه البرمجية
- إجراء الاختبار المكتبي لصحة الخوارزم

1- ضع خوارزما يستحث المستخدم على إدخال ثمن سلعة والكود الخاص بالسعر، ثم يحسب الخصم عليها بناء على الكود، ويطبع السعر الأصلي والخصم والسعر بعد الخصم. وفي حالة

عدم الخصم تخرج العبارة "لا خصم"، وإذا كان الحرف المدخل غير متضمن في قائمة الكـــــود، تخرج رسالة "كود خاطئ".

Design a program which will prompt the operator for the price of an article and a pricing code, and calculate the discount and print to the screen the original price, the discount, and the new price. The pricing code is as follows:

Pricing code	Discount rate
H	50%
F	40$
T	30%
Q	20%
Z	0%

If the pricing code is Z, the words "No discount" are tot be printed, if the code is not H,Q,T,F, or Z, the message "Invalid code" is to be printed.

ضع خوارزما يستحث المستخدم على إدخال رقم المسلسل للطلبة ودرجاتهم محسوبة مـــن 100، ويقابل كل درجة بالمستوى معبرا عنه بالحروف، ويخرج ذلك التقييم على الشاشة.

Design a program which will prompt the user for students serial number, and exam score out of 100, then match the scores to a letter grade and prints the grade. The letter grade is:

Exam score	Grade
90 and above	A
80-89	B
70-79	C
60-69	D
below 60	F

3- المطلوب خوارزم لبرنامج يقرأ عددين وكود من 1-4، فإذا كان الكود 1 يجمع العددين، 2 يطرح الثاني من الأول، 3 يضربهما، 4 وليس الثاني صفرا يقسم الأول على الثاني. يخرج البرنامج العددين، والكود ونتيجة الحساب.

Design a program which will read two numbers and an integer code form the screen. The value of the code is 1,2,3, or 4. If the code is 1 compute the sum, if 2 the difference (first minus second), 3 the product, 4 and the second is not 0, the quotient (first divided by second), print the two integers, code and result.

4- يحتوي ملف خاص بالعمولات على سعر البيع، كود المعاملة يحدد العمولة، ورقم البائع. ضع خوارزما يقرأ سجلا من سجلات الملف ويحسب العمولة المستحقة، ويخرج السعر والعمولة ورقك البائع.

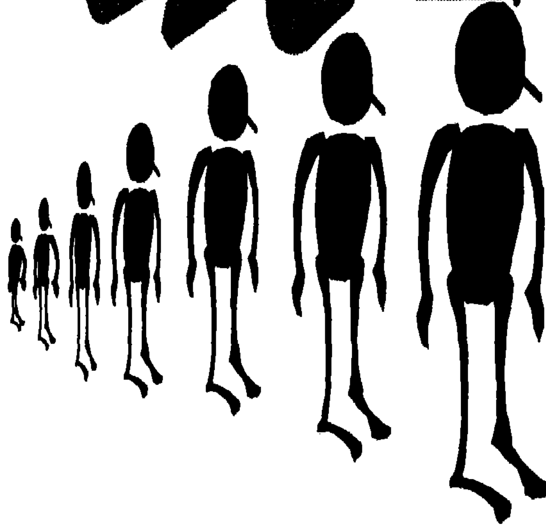
A transaction record on a Sales Commission File contains the retail price of an item sold, a transaction code which indicates the sales commission category to which item can belong, and the employee number of the person who sold the item. The transaction code can contain the values S, M, or L, which indicate that the percentage commission will be 5%, 7%, or 10% respectively. Construct an algorithm which will read a record on the file, calculate the commission owing for that record and print the price, commission and employee number.

5- شركة تسترد الزجاجات الفارغة لمنتجاتها بمقابل، في عبوات كل عبوة تحتوي على 10 زجاجات. وتريد وضع برنامج يتقبل عدد العبوات على صورة كسرية (مثلا؛ 8,5 يعني ثمانية عبوات كاملة ونصف عبوة)، وإذا زادت العبوات عن 8 حوسب العميل على 4 جنيه للعبوة، وإذا قلت عن ذلك حوسب على 3,5 للعبوة. يخرج البرنامج اسم العميل وعدد العبوات المعادة والمبلغ المستحق.

A glass return company requires a program which calculate the amount of credit due to a customer who returns cases of empty bottles (One case contains 10 bottles). Input to the program is a record containing the customer's name and an number, this number

contains the number of full cases or partly full cases (e.g. 8.5 means 8 full cases and one half-full case)> If 8 or more cases are returned the customer is to receive $4.00 per case, otherwise he receives $3.50 per case. The program is to print the customer's name, the number of cases returned, and the credit amount due.

الفصل الخامس هياكل التكرار



الأهداف:

- وضع خوارزم للهياكل DOWHILE و REPEAT...UNTIL
- وضع خوارزمات تحتوي على أنواع مختلطة من الدوارات
- عرض لمفهوم المصفوفات

الخطوط العامة :

1-5 التكرار باستخدام الهيكل DOWHILE

الخوارزمات التي عرضنا لها إلى الآن تشترك في أن كل أمر فيها ينفذ مرة واحدة، على أن أغلب البرامج تحتاج إلى تكرار عدد معين من الأوامر، والحل المتبع في هذه الحالة هو استخدام هيكل الدوارات.

وقد قدمنا لهيكل العبارة DOWHILE في الفصل الثاني، وبينا أن صورته العامة هي:

```
DOWHILE condition p is true
        statement block
ENDDO
```

وبما أن هذه الدوارة هي دوارة متقدمة leading decision loop فإن تنفيذها يسير على الوجه التالي:

- يختبر الشرط المنطقي
- إذا كان الشرط متحققا، سيتم تنفيذ بلوك العبارات مرة
- يعاد التحكم للدوارة، لتعاد الخطوة
- بمجرد أن يحدث عدم تحقق للشرط، ينتقل التحكم في تنفيذ البرنامج إلى العبارة التالية للكلمة الحاكمة ENDDO خارج الدوارة.

وهناك اعتباران هامان متعلقان بتصميم الدوارة DOWHILE: الأول؛ هو أن الاختبار يجري في بداية تشغيل الدوارة، وقد يترتب على ذلك استلزام إجراء بعض العمليات الأولية لوضع الشرط في صورة مناسبة للاختبار.

أما الأمر الثاني فهو أن الطريقة الوحيدة للخروج من الدوارة هو عدم تحقق الشرط، ويعني ذلك أنه يجب التأكد من وجود حالة لابد من حدوثها إن آجلا أو عاجلا، يكون الشرط فيها غير متحقق، وإلا دخل البرنامج في دوارة لانهائية، واستحال على البرنامج الخروج من الدوارة [ليس من حل في هذه الحالة إلا إعادة تشغيل الجهاز، ثم عدم تشغيل البرنامج إلا بعد التأكد من إزالة الخطأ، والبرامج المحتوية على دوارة لانهائية من الأخطاء الشائعة بين المبتدئين في البرمجة.

مثال 5-1 تحويل درجة الحرارة

لدينا محطة رصد لدرجات الحرارة تتلقى يوميا خمسة عشر درجة بالمقياس الفهرنهيتي، ويراد وضع برنامج لتحويلها إلى درجات حرارة مئوية، وإظهارها على شاشة الجهاز، ثم إظهار رسالة تفيد الانتهاء من عملية الرصد.

Every day, a weather station receives 15 temperatures expressed in degrees Fahrenheit. A program is to be written which will accept each Fahrenheit temperatures, convert it to Celsius and display the converted temperatures to the screen. After 15 temperatures has been processed, the message "All temperatures has been processed" is to be displayed on the screen.

A- Defining diagram

Input	Processing	Output
f_temp	Get Fahreheit temperatures	c_temp
(15 temperatures	Convert temperatures	
	Display Celius temperatures	
	Display message	

كالعادة، لا يتعرض جدول تحليل المسألة لكيفية التحويل، فهو يهتم بما هو مطلوب وليس بكيفية تنفيذه. وبعد أن تعرفنا على العناصر الثلاثة للبرنامج: المدخلات والمخرجات وعمليات المعالجة، نشرع في وضع حل المسألة، ويمكننا القيام بذلك بتسجيل الهياكل المطلوبة للتنفيذ، وأية متغيرات أخرى قد يتطلبها الخوارزم، وفي مثالنا هذا نحن محتاجون إلى:

- هيكل تكرار لعبارة DOWHILE
- عداد *counter* يستهل بالصفر لمتابعة عملية التكرار، وهو متغير سوف نسميه temp_counter

والآن، لنضع معا الخوارزم

B- Solution algorithm

```
Fahrenheit_Celcius_conversion
    Set temp_counter to zero
    DOWHILE temp_counter <15
        Prompt operator for f_temp
```

```
        Get f_temp
        Compute c_temp = (f_temp - 32) * 5/9
        Display c_temp
        Add 1 to temp_counter
    ENDDO
    Display 'All temperatures has been processed'
END
```

وعليك أن تلاحظ أن العداد temp_counter قد استهل بالصفر قبل بداية الدوارة، وأنه قـد اختبر في بدايتها، ويجري عليه عملية تزايد خلال تشغيلها. ومن المهم الانتباه إلى أن المتغير الـذي يتابع عدد مرات التكرار يكون دائما تحت التحكم بهذه الصورة في هـذه المواضـع الثلاثـة. وتلاحظ أيضا أن العبارة التي تقوم بتزايد العداد هي آخر عبـارة في بلـوك الـدوارة [قبـل ENDDO]، بمعنى أنه بعد التزايد مباشرة تجري عملية اختبار العداد من جديد.

ومن الممكن صياغة الخوارزم باستخدام WHILE...DO وENDWHILE على الصـورة التالية:

```
Fahrenheit_Celcius_conversion
    Set temp_counter to zero
    WHILE temp_counter <15 DO
        Prompt operator for f_temp
        Get f_temp
        Compute c_temp = (f_temp - 32) * 5/9
        Display c_temp
        Add 1 to temp_counter
    ENDWHILE
    Display 'All temperatures has been processed'
END
```

C- Desk checking

على الرغم من أن البرنامج يتضمن تكرار عملية التحويل خمسة عشر مـرة، إلا أنـه في هـذه المرحلة يمكن الاكتفاء بمجموعتين من المدخلات كالمعتاد.

i- Input data

	First data set	Second data set
f_temp	32	50

ii- Expected results

	First data set	Second data set
c_temp	0	10

iii- Desk check table

Statement	temp_ counter	DOWHILE condition	f_temp	c_temp
Initialize	0			
DOWHILE		true		
Get			32	
Compute				0
Display				yes
Add	1			
DOWHILE		true		
Get			50	
Compute				10
Display				yes
Add	2			

وفي الاختبار المكتبي لهذا الخوارزم علينا أن نبين عمل الدوارة واضحا، فنبين أولا خطوة الاستهلال، ثم اختبار الشرط، ثم التزايد، فاختبار الشرط بعد ذلك مباشرة، على الوجه الذي شرحناه آنفا. كما يتضح لنا أن الدوارة لابد آتية إلى خطوة يكون الشرط فيها غير متحقق، حيث تصل الخطوة Add إلى 15، وبالتالي فلا يوجد خطر الدوارة اللانهائية.

مثال 5-2 طباعة نتائج اختبار-معالجة البيانات التتابعية

لنأخذ الآن حالة لا يكون فيها عدد التكرار معروفا من البداية، فلا يكون هناك جدوى من استخدام العداد لتحديد وقت الخروج من الدوارة، ومن ثم فلابد من اتخاذ وسيلة أخرى تمكن المستخدم من الخروج منها، كما يتضح من المثال التالي.

يراد تسجيل نتائج اختبار مادة ما، والمطلوب برنامج يتقبل اسم الطالب، ودرجته، ويظهرهما على الشاشة، ثم يقوم بحساب متوسط الدرجات في نهاية التسجيل، والدرجات تتراوح بين الصفر و100.

ولما كان عدد الطلاب غير معروف لمدخل البيانات مسبقا، فإننا سوف نعطي تعليمات له أن يتصرف كما يلي عند الانتهاء من تسجيل كافة الطلبة: يترك خانة الاسم للسجل التالي شاغرة، ويضع في خانة الدرجات الرقم 999، فيكون هذا الرقم في خانة الدرجات هو الشرط الذي تنتهي عنده الدوارة. ويطلق على مثل هذا السجل "السجل الرقيب sentinel أو سجل التذييل "trail record

A program is required to read and print a series of names and exam scores for students, the class average is to be computed and printed at the end of the report. Scores can range from 0 to 100. The last record contains a blank name and a score of 999 and should not be included in the calculation.

A- Defining diagram

Input	Processing	Output
name	Read student detail	name
exam_score	Print student detail	exam_score
	Compute average score	avg_score
	Pront average score	

ولوضع خوارزم الحل فأنت محتاج لما يلي:

- هيكل دوارة DOWHILE تتحكم في سير البرنامج إلى أن يدخل العدد 999 في خانة الدرجات.
- متغير تتراكم فيه الدرجات للحصول على مجموعها (مركم **accumulator**)، سوف نسميه totl_score.
- مركم يتراكم فيه عدد الطلبة للحصول على مجموعها، سوف نسميه total_students.
- يستهل المركمان بقيمة صفرية في بداية البرنامج

وإليك خوارزم الحل:

B- Solution algorithm

```
Print_exam_scores
    Set total_score to zero
    Set total_students to zero
    Read name, exam_score
    DOWHILE exam_score not = 999
      Add 1 to total_students
      Print name, exam_score
      Add exam_score to total_score
      Read name, exam_score
    ENDDO
    IF total_students not = 0 THEN
      avg_score = total score/total_students
      Print avg_score
    ENDIF
END
```

ويمثل هذا الخوارزم مثالا نمطيا لتصميم برنامج يتولى معالجة ملف ذي بيانات تتابعية sequential. وفي مثل هذا النوع من البرامج، يجب قراءة سجل واحد قبل الدخول في الدوارة، وإجراء اختبار استمرارها، ويتبين ذلك من وجود السطر Read name, exam_score: قبل الدخول في الدوارة مباشرة. وتسمى هذه العملية "القراءة التمهيدية priming read" وهي غاية في الأهمية في معالجة البيانات التتابعية.

ويحتاج الخوارزم إلى عملية قراءة أخرى، ولكنها توجد هذه المرة داخل بلوك الدوارة ذاتها. والموضع الذي توجد به هام أيضا، إذ يجب أن يتلوها تكرار الدوارة على الفور، حتى نضمن ألا يدخل الرقم 999 في حساب المتوسط. ولهذا تجد أن موضعها قبل عبارة إنهاء الدوارة ENDDO، بحيث يتلو تنفيذها إعادة اختبار شرط استمرارية الدوارة، وهو ألا تكون الدرجة المدخلة 999. وفي اللحظة التي يدخل فيها هذا الرقم، ينتقل البرنامج إلى حساب المتوسط، فلا تسجل هذه الدرجة.

والقراءة التمهيدية قبل الدوارة، مع قراءة داخلها، قبل عبارة إنهائها مباشرة، يمثـــلان الهيكـــل الأساسي لخوارزم الدوارة التي تستخدم لمعالجة البيانات التتابعية، والتي تكون صورته العامة على النحو التالي:

```
Process_sequential_file
    initial processing
    Read first record
    DOWHILE more records exist
       Process this record
       Read next record
    ENDDO
    final processing
END
```

C- Desk checking

نضيف على مجموعتي المدخلات التجريبية المعتادة مدخلا يمثل شرط الخروج من الدوارة

i-Input data

	First data set	Second data set	Third data set
score	50	100	999

تتمثل النتائج المتوقعة فيما يلي:

الاسم الأول: النتيجة 50

الاسم الثاني: النتيجة 100

المتوسط: 75

ii- Expected result
1st name and score of 50
2nd name, and score of 100
Average score 75

iii- Desk chek table

Statement	total_ score	total_students	exam_ score	DOWHILE condition	avg_score
Initialize	0	0			
Read			50		
DOWHILE				true	
Add		1			
Print			yes		
Add	50				
Read			100		
DOWHILE				true	
Add		2			
Print			yes		
Add	150				
Read			999		
DOWHILE				false	
Compute					75
Print					yes

مثال 5-3 معالجة تسجيل الطلاب

يراد وضع برنامج يقرأ أسماء الطلاب، وينتخب منهم المقيدين في مادة "برمجة 1". ويحتوي سجل كل طالب على رقمه، واسمه، وعنوانه، والرقم البريدي، والجنس، والرقم الكودي للمادة، وهو بالنسبة للمادة المطلوبة 18500. وفي نهاية التسجيل يراد إخراج ثلاثة مجاميع: مجموع الفتيات، ومجموع الفتيان، ومجموع الطلبة.

A program is required which will <u>read</u> a file of students records, and <u>select</u> and <u>print</u> only those students enrolled in a unit named Programming 1. Each student record contains student number, name, address, postcode, gender and course unit number. The course unit number for Programming 1 is 18500. Three totals are to be <u>printed</u> at the end of report; total females, total males and total students.

A- Defining diagram

Input	Processing	Output
student_record	Read student records	selected students
. student_no	Select student records	records
. name	Print selected records	totals
. address	Compute total females	
. postcode	Compute total males	
. gender	Compute total students	
. course_unit	Print totals	

لوضع خوارزم هذا البرنامج، عليك أن تتدبر الأمور التالية:

- استخدام الدوارة
- استخدام عدة عبارات شرطية
- استخدام عدد من المراكم لتجميع المجاميع (ثلاثة مراكم)

وبالنسبة للأمر الأول، تلاحظ أن شرط استمرارية الدوارة في هذا المثال سيكون هو عدم الوصول إلى نهاية الملف، ويتعارف على إعطائه الرمز EOF والذي يرمز إلى العبارة end of file، أي أن شرط الاستمرارية سوف يعبر عنه بالصورة:

DOWHILE not EOF

على أنه يمكن استخدام أي تعبير عن مثل هذا الشرط، فالتعابير التالية كلها تؤدي الغرض:

DOWHILE more data
DOWHILE more records
DOWHILE records exist
WHILE success DO

وفي هذه الحالة سوف يوكل للبرنامج تحديد وقت الخروج من الدوارة، حينما لا يوجد مزيد من السجلات يقرأها، ويخبره بذلك تحول الشرط من حالة التحقق إلى حالة عدم التحقق.

B- Solution algorithm

```
Process_students_enrolments
      Set total_fem to zero
      Set total_mal to zero
      Set total_std to zero
```

```
        Read student record
        DOWHILE not EOF
            IF course_unit = 18500
                print student details
                increment total_std
                IF std_gndr = female THEN
increment total_fem
                ELSE
increment total_mal
                ENDIF
            ENDIF
            read student record
        ENDDO
        Print total_fem
        Print total_mal
        Print total_std
END
```

C- Desk checking

يكفي اختبار البرنامج بثلاثة مدخلات تجريبية. وحيث إن رقم الطالب واسمه وعنوانه والرقم البريدي ليست جميعا موضوع معالجة في الخوارزم، فيكتفى برقم المادة، والذي يحدد السجلات التي سوف يتم معالجتها.

	First data set	Second data set	Third data set
course unit	20000	18500	18500
gender	F	F	M

ii- Expected results:

Student number, name, address, postcode, F (2nd student)
Student number, name, address, postcode, M (3rd student)

Total females1
Total males1
Total students2

iii- Desk checking table

Statement	course_ unit	gender	DOWHILE condition	IF condition	total_ fem	total_ mal	total - std
Initialize					0	0	0
Read	200000	F					
DOWHILE			true				
IF				false			
Read	18500	F					
DOWHILE			true				
IF	Print	Print		true			1
IF				true	1		
Read	18500	M					
DOWHILE			true				
IF	Print	Print		true			2
IF				false		1	
Read	EOF						
DOWHILE				false			
Print					yes	yes	yes

2-5 الهيكل التكراري باستخدام REPEAT....UNTIL

يختلف هذا الهيكل عن الهيكل السابق في كون الأخير يختبر شرط الاستمرارية في بداية الـدوارة، بينما في هيكلنا هذا يختبر شرط إنهائها في نهاية الدوارة، الأمر الذي يعني أن بلوك الدوارة سوف ينفذ مرة على الأقل، ثم يختبر الشرط، ويظل التكرار منفذا طالما كان شرط الإنهاء غير متحقـق. وعلى ذلك؛ فالصورة العامة لهذا الهيكل كالتالي:

```
REPEAT
statement
statement



UNTIL condition is true
```

وتسمى هذه الدوارة "دوارة متأخرة الاختبار trailing decision loop"، وهـي عكـس الدوارة متقدمة الاختبار السابقة. ويترتب على اختلاف طبيعتي نوعـي الدوارتـين اختلافـان

آخران: الأول هو أن شرط استمرارية الدوارة هنا هو في هذه الدوارة هو عدم التحقق وليس التحقق، وبذلك فإن الشرطان منعكسان منطقيا في الدوارتين، فبينما يكون الشرط في الدوارة متقدم الاختبار هو: DOWHILE more records مثلا؛ يكون في الأخرى:REPEATE.....UNTIL no more records، وإذا كان في الأولى DOWHILE not EOF يكون في الثانية REPEATE.....UNTIL EOF.

أما الاختلاف الثاني فهو أن الدوارة متأخرة الاختبار ليست محتاجة لقراءة تمهيدية، حيث إن بلوك الدوارة سوف ينفذ مرة على الأقل عند الدخول فيها.

وللمقارنة بين النوعين، لنأخذ الخوارزم التالي:

```
Process_record_students
      Set student_counter to zero
      Read student record
      DOWILE student number NOT =999
         Write student record
         Increment student_count
         Read student record
      ENDDO
print student_count
END
```

ولنقارن بينه وبين الخوارزم التالي:

```
Process_record_students
      Set student_counter to zero
      REPEAT
      Read student record
         Write student record
         Increment student_count
         Read student record
      UNTIL student_record = 999
print student_count
END
```

وتدل النظرة الفاحصة على أن الخوارزمين ليسا متطابقين كما قد يبدو للوهلة الأولى، ذلك لأنه بعد قراءة الملف الرقيب (عدم تحقق شرط الاستمرارية) يستمر تنفيذ البلوك مما يؤدي لنتائج خاطئة، حيث سيؤخذ بيانات هذا السجل في الاعتبار، بينما المفروض هو إنهاء الدوارة فور الوصول إليه.

وللتغلب على هذا العيب يضاف شرط داخل الدوارة يمنع معالجة السجل الأول لو كان غير متفق مع الشرط، على الوجه التالي:

```
Process_record_students
      Set student_counter to zero
      REPEAT
      Read student record
         IF student_mumber NOT = 999 THEN
         Write student record
         Increment student_count
         Read student record
         ENDIF
      UNTIL student_record = 999
print student_count
END
```

والشرط الذي تمت إضافته هو:

```
IF student_mumber NOT = 999 THEN
```

وهو يضمن عدم تنفيذ الدوارة عند إدخال رقم 999 في السجل

ولهذا السبب تجد أن هذا الهيكل أقل شيوعا عن الأول، وهو ما سوف نتبعه في برامجنا التالية بالفعل، إلا أن مثالنا التالي سوف يكون عن هذه الدوارة.

مثال 5-4 معالجة أصناف مخزنية

يراد وضع برنامج يسجل الأصناف المخزنية، يحتوي كل سجل على بيانات هي رقم الصنف، واسم الصنف، والرصيد المخزني. وقد أعطي آخر سجل في الملف رقم صفر (السجل الرقيب).

والمفروض أن يخرج البرنامج تقريرا بالأصناف التي يقل مخزونها عن 20 وحدة، على أن يحتـوي التقرير على مقدمة، كما يخرج مجموع هذه الأصناف في نهايته.

A program is required to read a series of inventory records that contain items number, item description and stock figure. The last record in the file has an item number 0. The program is to produce a "Low Stock Items" report, by printing only those records which have stock figure less than 20 items. A heading is to be printed at the top of the report and a total low stock item count to be printed at the end.

A- Defining diagram

Input	Processing	Output
Inventory record	**Read inventory records**	**heading**
. item_number	**Select low stock items**	**selected records**
. item_descrip	**Print low stock records**	**. item_number**
. stock_figure	**Print total low stock records**	**. item_descrip**
		. stock_figure
		total_low_stock

لوضع الخوارزم، عليك أخذ الآتي في الاعتبار:

الاحتياج للدوارة (أي من النوعين سيفي بالغرض)

الاحتياج لعبارات شرطية لتحقيق الانتخاب للسجلات

الاحتياج لمركم للتجميع

في حالة استخدام دوارة متأخرة الاختبار، الحاجة لشرط إضافي يمنع معالجـة أي سـجل غـير مطلوب.

وإليك الخوارزم الذي يستخدم هذه الدوارة:

B1- Solution algorithm

```
Process_inventory_records
    Set total_low_stock to zero
    Print 'Low Stock Items' heading
    REPEAT
      Read inventory record
      IF item_number > zero THEN
        IF stock_figure < 20 THEN
```

```
            print item_number, item_descrip, stock_figure
increment total_low_stock
            ENDIF
        ENDIF
    UNTIL intem_number = zero
    Print total_low_stock
END
```

وإليك الخوارزم الذي يستخدم الدوارة متقدمة الاختبار:

B2- Solution algorithm

```
Process_inventory_records
    Set total_low_stock to zero
    Print 'Low Stock Items' heading
    Read inventory record
    DOWHILE item_mumber > zero
        IF stock_figure < 20 THEN
            print item_number, item_descrip, stock_figure
            increment total_low_stock
        ENDIF
        Read inventory record
    ENDDO
    Print total_low_stock
END
```

سوف يجرى الاختبار المكتبي على خوارزم الدوارة الأولى.

C- Desk checking

سوف نأخذ مجموعتين من المدخلات التجريبية، بالإضافة إلى السجل الرقيب (حيث رقم الصنف = صفر) للاختبار.

i- Input data

	First data set	Second data set	Trailer record
item_number	123	124	0
stock_fig	8	25	

ii- Expected results

Low Stock items

123 8(first record)

Total Low Stock Items = 1

iii- Desk check table

Statement	Item_number	stock_fig	REPEAT UNTIL	first IF condition	second IF condition	total_low_stock	heading
Initialize						0	
Print							yes
Read	123	8					
IF				true			
IF	Print	Print			true	1	
UNTIL			false				
Read	124	25					
IF				true			
IF					false		
UNTIL			false				
Read	0						
IF				false			
UNTIL			true				
Print						yes	

3-5 هياكل التكرار المحسوب

إذا كان عدد التكرار معروفا مسبقا، فيمكن استخدام هيكل أكثر بساطة مما عرضنا له سابقا، يستخدم كلمة DO مع دليل للدوارة loop_index يحدد العددين الابتدائي initial_value والنهائي final_value بحيث تكون الصورة العامة للهيكل على الوجه التالي:

```
DO loop_index = initial_value to final_value
    statement block
ENDDO
```

ويحقق هذا الهيكل ما هو أكثر من مجرد التكرار، فهو يتولى:

- استهلال دليل الدوارة بالقيمة المطلوبة في الابتداء
- تزايد الدليل مع التكرار
- اختبار قيمة الدليل مع كل تكرار لمعرفة الوصول للعدد النهائي من عدمه
- انهاء الدوارة بمجرد الوصول للعدد النهائي

وبعبارة أخرى، فهذا الهيكل يحقق كل ما هو مطلوب للدوارة، الاستهلال والتزايد والاختبـــار بصورة آلية، وأيضا إنهاؤها عندما تستنفذ عدد مرات التكرار.

وسوف نعيد مثال 5-1 مستخدمين هذه الفكرة الجديدة.

مثال 5-5 تحويل درجات الحرارة

Every day, a weather station receives 15 temperatures expressed in degrees Fahrenheit. A program is to be written which will accept each Fahrenheit temperatures, convert it to Celsius and display the converted temperatures to the screen. After 15 temperatures has been processed, the message "All temperatures has been processed" is to be displayed on the screen.

A- Defining diagram

Input	Processing	Output
f_temp	Get Fahreheit temperatures	c_temp
(15 temperatures	Convert temperatures	
	Display Celius temperatures	
	Display message	

سوف يحتاج خوارزم الحل إلى دوارة DO مع عداد هو temp_counter ليتابع عدد التكرار.

B- Solution algorithm

```
Fahrenheit_Celcius_conversion
      DO temp_counter= 1 to 15
         Prompt operator for f_temp
         Get f_temp
         Compute c_temp = (f_temp - 32) * 5/9
         Display c_temp
      ENDDO
      Display 'All temperatures has been processed'
END
```

ومن ذلك ترى أن الهيكل يتحكم في كافة الخطوات المطلوبة للدوارة:

- استهلال قيمة العداد temp_counter بـــ 1
- تزايد العداد مع كل مرة تكرار

- اختبار قيمة العداد في بداية كل دورة
- إنهاء الدوارة آليا عند وصول العداد للعدد النهائي

C- Desk checking

i- Input data

	First data set	Second data set
f_temp	32	50

ii- Expected results

	First data set	Second data set
c_temp	0	10

iii- Desk check table

Statement	temp_counter	DO	f_temp	c_temp
DO	1	set to 1		
Get			32	
Compute				0
Display				yes
DO	2	increment		
Get			50	
Compute				10
Display				yes

4-5 المصفوفات

المصفوفات هي إحدى أشكال هياكل البيانات data structures تتكون من عدد من المتغيرات من نفس النوع، مثلا؛ درجات مجموعة من الطلبة. ويكون للمصفوفة اسم موحد، وليكن scores، ويعرف كل متغير في المصفوفة عن طريق موضعه فيها، ويتم ذلك باستخدام دليل المصفوفة array_index، يلحق بالمصفوفة كالتالي scores(3) أو $scores_3$ للتعبير عن ثالث عنصر في المصفوفة.

ويتم التعامل مع عناصر المصفوفات بالتتابع، وهو ما يمكنك عمله بسهولة بواسطة هيكل التكرار DO كما نعرض في المثال التالي.

مثال 5-6 تسجيل درجات طلاب

يراد وضع برنامج يتلقى درجات ثمانية عشر من الطلاب ويحسب متوسط الدرجات ويظهر النتيجة على الشاشة.

Design a program which will prompt for and receive 18 exam scores, compute the class average, and display all the scored and the average on the screen.

A- Defining diagram

Input	Processing	Output
scores	Prompt for scores	scores
	Get scores	avg_scores
	Compute avg_scores	
	Display scores	
	Display avg_scores	

هذا مثال لهيكل تكراري يعلم مسبقا عدد التكرارات (18)، وسوف نحتاج لوضع الخوارزم إلى:

- مصفوفة لتخزين الدرجات، وهي المصفوفة scores
- دليل يدل على مواضع عناصر المصفوفة، وهو index
- مركم لتجميع الدرجات، وهو total_score
- دوارة من نوع DO لتلقي الدرجات
- دوارة أخرى من نفس النوع لإظهار الدرجات

ويكون الخوارزم على الصورة التالية:

B- Solution algorithm

```
Process_exam_scores
    Set total_scores to zero
    DO index = 1  to 18
       Prompt for scores
```

```
Get scores(index) (أمر تلقي عناصر المصفوفة واحدا بعد الآخر)
        total_scores = total_score + scores(index)
     ENDDO
     compute avg_score = total_scores/18
     DO index = 1  to 18
        display scores(index)
        ENDDO
     Dislplay avg_scores
END
```

C- Desk checking

i- Input data

	Input data set
scores	18 scores of 100

ii- Expected result

	Input data set
avg_scores	100

iii- Desk check table

Statement	total_ score	first DOloop	second doloop	avg_score	scores
Initialization	0			.	
DO	1800	OK			100x1 8
Compute				100	
DO			OK		displa yed
Display				yes	

5-5 ملخص الفصل

غطينا في هذا الفصل هيكل التكرار بالتفصيل، مع الأمثلة على كل من الدوارتين DOWHILE وREPEAT....UNTIL. كما تعرضنا أيضا للتكرار المحسوب، وقدمنا العديد من الخوارزمات التي تستخدم هياكل التحكم الثلاثة. وأخيرا قدمنا للمصفوفات وضربنا مثالا يستخدم التكرار المحسوب للتعامل معها.

وقد بينا أن أغلب الخوارزمات لها نمط متشابه:

1- بعض العمليات قبل الدخول فيها؛

2- بعض العمليات داخلها؛

3- بعض العمليات بعد الخروج منها.

وعلى ذلك فإن الصورة العامة لخوارزم الدوارة DOWHILE مثلا يكون كما يلي:

```
Process_sequential_file
        Initial processing
        Read first record
        DOWHILE more records exist
                Process this record
                Read next record
        ENDDO
END
```

5-5 تدريبات

في المسائل الآتية مطلوب منك:

- تحليل المسألة عن طريق جدول التحليل
- وضع الخوارزم بأسلوب اللغة شبه البرمجية
- إجراء الاختبار المكتبي لصحة الخوارزم

1- صمم خوارزما يستحث ويتقبل ويحسب مجموع أجور لعاملين، إلى أن يدخل قيمة -99 (سجل رقيب)، بعدها يخرج البرنامج المجموع.

Design a program which prompts, receives and calculates total of a collection of payroll amounts until a sentinel amount of -99 is entered then display the total.

2- صمم خوارزما يقرأ عددا من الأعداد الصحيحة من جهاز الإدخال، أول رقم فيها يحدد عدد الأعداد التالية، ويحسب مجموع ومتوسط هذه الأعداد عدا الأول منها ويظهرهما على الشاشة.

Design an algorithm that will read a series of integers at the terminal. The first integer is a special one, as it will indicate how many more integers will follow. Your algorithm is to compute and print the sum and average of the integers, excluding the first one, and displays them to the screen.

3- صمم خوارزما يعالج قسيمة ساعات العمل، تحتوي على: رقم العامل، ساعات العمل في الأسبوع، والأجر في الساعة. تحسب الساعات الإضافية التي تزيد عن 35 ساعة بواقع مرة ونصف الأجر المعتاد، ويخرج البرنامج رقم العامل وأجره المستحق بعد خصم ضرائب بواقع 15%. وفي نهاية المعالجة يظهر البرنامج مجموع الأجور ومتوسطها.

Design an algorithm that processes the weekly employees time card for all employees. Each card contains: employee's number, hourly wage rate and the number of hours worked during the week. The employee is paid time-and-half for overtime over 35 hours. a tax of 15% is deducted of gross salary.The output should display the employee's number and net pay. At the end of the run, display the total payroll amount and the average net amount paid..

4- يراد تصميم برنامج يقرأ ملف أجور تحتوي سجلاته على رقم العامل، اسم العامل، الأجر في الساعة، ساعات العمل العادية والإضافية، ويحسب ويظهر الأجر الإجمالي في تقرير بعنوان "تقرير الأجر الأسبوعي" مبينا به أيضا كل البيانات المدخلة. .

Design a program which reads a file of employee records containing the employee's number, name, hourly pay rate, regular hours worked

and overtime and computes the total. pay on the "Weekly Payroll Report". All input data should appear on the report.

5- يراد تصميم برنامج يقرأ إنتاج مصنع من سلع معينة من ملف تحتوي سجلاته علـــى رقـــم المنتج واسمه وعدد القطع المنتجة في هذا العام والعام الماضي، ويراد إظهار تقرير يحوي رقم المنتج واسمه والزيادة في الإنتاج أو النقص فيه من كل صنف.

Design a program which will read a file of product records, each containing the item number, name, quantity produced this year and last year. The program should produce a "Product List" showing the item number, name and the increase or decrease in the quantity produced.

الفصل السادس تطبيقات



الأهداف:

- أمثلة تطبيقية على وضع الخوارزمات

الخطوط العامة :

6-1 ثمانية أمثلة تطبيقية

6-2 تدريبات

مثال 6-1 معالجة أزواج من الأعداد

ضع برنامجا يتقبل عددين، ويخرج مجموعهما، وحاصل ضربهما، ومتوسطهما الحسابي، وإذا كان المجموع أكبر من 200، يخرج البرنامج نجمة بجوار المجموع. ينتهي البرنامج عند إدخال زوج من صفرين.

Design a program which will prompt for and receives pairs of numbers, display their sum, product, and average on the screen. If the calculated sum is over 200 then an asterisk is to be displayed beside the sum. The program is to terminate when a pair of zero values is entered.

A- Defining diagram

Input	Processing	Output
number_1	Prompt for numbers	sum
number_2	Get two numbers	product
	Calculate sum	average
	Calculate product	'*'
	Calculate average	
	Display sum, product, average	

B- Control structured required

1. A DOWHILE loop to control the repetition.
2. An IF statements to determine if an astrik is to be displayed
3. Note the use of the NOT operand with the AND operator.

C- Solution algorithm

```
Process_number_pair
      Set sum to zero
      prompt for number_1, number_2
      Get number_1, number_2
      DOWHILE NOT (number_1 = 0 AND number_2 = 0)
         sum = number_1+ number_2
         product = number_1* number_2
         average = sum/2
         IF sum > 200 THEN
```

```
            Display sum, '*', product, average
        ELSE
            Display sum, product, average
        ENDIF
        prompt for number_1, number_2
        Get number_1, number_2
    ENDDO
END
```

مثال 6-2 طباعة سجلات طلبة

في ملف للطلبة يوجد نوعان من السجلات 'S' و'U'، الأول يحتوي على بيانات الطالب الشخصية: الرقم والاسم والعنوان والجنس ونظام الالتحاق (منتظم أو غير منتظم)، ويحتوي الثاني على رقم المادة أو المواد المسجل لها الطالب. المطلوب برنامج يقرأ هذه الملفات ويستخرج منها بيانات الرقم والاسم والعنوان في ملف باسم "قائمة الطلاب".

A file of student records consists of 'S' records and 'U' records. An 'S' record contains the student's name, age, gender and attendance pattern; full time (F/T) or part time (P/T). A 'U' record contains the number and name of the unit or units in which the student has enrolled. There may be more than one 'U' record for each 'S' record. Design a solution algorithm which will read the file of student records and print only the student's numbers, name and address ro n "STUDENT LIST".

A- Defining diagram

Input	Processing	Output
'S' records	Print heading	heading of file
. number	Read student records	selected student records
. name	Select 'S' records	. number
. address	Print selected records	. name
. gender		. address
. age		
. attendance		

Input	Processing	Output
'U' records		

B- Control structures required

1.A DOWHILE loop to control the repitition
2.An IF statement to select 'S' records

C- Solution algorithm

```
Print_student_records
    Print 'STUDENT LIST'
    Read student record
    DOWHILE more records exists
        IF student record = 'S' record THEN
            print student_number, name, and address
        ENDIF
    ENDDO
END
```

مثال 6-3 طباعة عدد مختار من الطلبة

في المثال السابق، المطلوب استخراج قائمة بالطالبات غير المنتظمات، يعنون "الطالبات غير المنتظمات"، وتضم الرقم والاسم والعنوان والسن.

Design a program which will read the same student file as in Example 6.2 , and produce a report of all female students who are enrolled part time. The report is to be headed 'PART TIME FEMALE STUDENTS' and is to show the student's number, name, address and age.

A- Defining diagram

Input	Processing	Output
'S' records	Print heading	heading of file
. number	Read student records	selected student records
. name	Select P/T female records	. number
. address	Print selected records	. name

Input	Processing	Output
. gender		. address
attendance		. age
. age		
'U' records		

B- Control structures required

1.A DOWHILE loop to control the repitition

2.An IF statement to select 'S' records

C- Solution algorithm

أ) الخوارزم باستخدام تداخل غير خطي لعبارات شرطية

```
Print_student_records
    Print 'PART TIME FEMALE STUDENTS'
    Read student record
    DOWHILE more records exists
      IF student record = 'S' record THEN
        IF attendance = P/T THEN
IF gender = female THEN
            print student_number, name, and address, age
          ENDIF
        ENDIF
      ENDIF
      read student record
    ENDDO
END
```

ب) الخوارزم باستخدام عبارات شرطية متداخلة ومركبة

```
Print_student_records
    Print 'PART TIME FEMALE STUDENTS LIST'
    Read student record
    DOWHILE more records exists
      IF student record = 'S' record THEN
        IF attendance = P/T THEN
        AND  gender = female THEN
```

```
        print student_number, name, and address, age
      ENDIF
    ENDIF
    read student record
  ENDDO
END
```

ج) الخوارزم باستخدام عبارات شرطية مركبة

```
Print_student_records
    Print 'PART TIME FEMALE STUDENTS LIST'
    Read student record
    DOWHILE more records exists
      IF student record = 'S' record THEN
      AND attendance = P/T THEN
      AND gender = female THEN
        print student_number, name, and address, age
      ENDIF
      read student record
    ENDDO
END
```

مثال 6-4 طباعة طلاب مختارين ومجموعهم

في المثال السابق، المطلوب استخراج نفس قائمة بالطالبات غير المنتظمات، ثم يضاف للتقرير عدد الطالبات وعدد مجموع الطلبة والطالبات.

Design a program which will read the same student file as in Example 6.3 , and produce the same 'PART TIME FEMALE STUDENTS' report. In addition, you are to print at the end of the report the number of selected students and the total number of students on the file.

A- Defining diagram

Input	Processing	Output
'S' records	Print heading	heading of file
. number	Read student records	selected student records
. name	Select P/T female records	. number
. address	Print selected records	. name
. age	Compute total students	. address
. gender	Compute total selected students	. age
attendance	print totals	toal_studs
'U' records		total_selected_stds

B- Control structures required

1. A DOWHILE loop to control the repitition
2. An IF statement to select 'S' records
3. Accumulator for total_stds and total_selected_stds

C- Solution algorithm

```
Print_student_records
        Print 'PART TIME FEMALE STUDENTS LIST'
        Set total_stds to zero
        Set total_selected_stds to zero
        Read student record
        DOWHILE more records exists
                IF student record = 'S' record THEN
                        increment total_stds
                        IF attendance = P/T THEN
                        AND  gender = female THEN
                                increment total_selected_stds
                        print student_number, name, and address, age
                        ENDIF
                ENDIF
                read student record
        ENDDO
        Print total_stds
```

```
    Print total_selected_stds
END
```

وفي مثل هذه البرامج يجب الانتباه جيدا إلى موضع تزايد المراكم، فالخطأ في ذلك ينتج نتائج غير سليمة.

مثال 6-5 معالجة مصفوفة أعداد صحيحة

يراد تصميم برنامج يقرأ مصفوفة من مائة عدد صحيح ويحسب عدد الأعداد التي يزيد قيمتـــها عن متوسط أعداد المصفوفة، ويظهر المتوسط وعدد الأعداد الأكبر من المتوسط.

Design an algorithm which will read an array of 100 integer values and count the number of integers of that array which are greater than the average values of all the integers in the array. The algorithm is to display the average integer value and the count of integers greater than the average.

A- Defining diagram

Input	Processing	Output
100 integer values	Read integer values	average
	Calculate average value	integer_count
	Compute count of selected integers	
	Display average value	
	Display selected integer count	

B- Control structures required

1. an array of integers values, i.e. numbers (اسم المصفوفة)
2. a DO loop to calculate the average of the integers, and
3. a DO loop to count the number of integers greater than the average value.

C- Solution algorithm

```
Process_integer_array
    Set integer_total to zero
    Set integer_count to zero
    DO index = 1 to 100
        integer_total = integer_total + number(index)
    ENDDO
```

```
        average = integer_total/100
        DO index = 1  to 100
           IF number(index) > average THEN
              add 1 to integer_count
           ENDIF
        ENDDO
        Display average, interger_total
END
```

مثال 6-6 إنتاج تقرير مبيعات

مطلوب برنامج يقرأ ملف مبيعات وينتج تقريرا لها. كل سجل في الملف يحتوي على رقم العميل، اسمه، مقدار المبيعات وكود الضرائب الخاص به. والمطلوب أن يطبع في مقدمة التقرير اسمه، ثم تفاصيل تضم رقم العميل، الاسم، مقدار المبيعات، ضريبة المبيعات وإجمالي القيمة.

Design a program which will read a file of sales records and produce a sales report. Each record contains customer's number, name, a sales amount and tax code as follows:

tax codesales tax
0tax exempt
13%
25%

The report is to print a heading "SALES REPORT", and detail lines listing the customer number, name sales amount, sales tax and the total amount due from the customer.

A- Defining diagram

Input	Processing	Output
sales records	Print heading	heading line
customer_number	Read sales records	detail lines
name	Calculate sales tax	cutomer_number
sales_amt	Calculate total amount	name
tax_code	Print customer details	sales_amt
		sales_tax
		total_amt

B- Control structures required

1.A WHILE... DO loop to control the repitition, and
2.A CASE statement to calculate the sales_tax

C- Solution algorithm

```
Process_sales_record
    Print "SALES REPORT" heading
    Read sales record
    WHILE success DO
        CASE of tax_code
            0: sales_tax = 0
            1: sales_tax = sales_amt*0.03
            2: sales_tax = sales_amt*0.05
        ENDCASE
        total_amt = sales_amt + sales_tax
        Print ustomer_number, name, sales_amt, sales_tax, total_amt
        Read sales record
    ENDWHILE
END
```

كان من الممكن استخدام العبارات الشرطية المتداخلة خطيا، ولكن عبارة CASE هي المحبذة في مثل هذه الأحوال.

مثال 6-7 نتائج اختبارات طلبة

يراد تصميم برنامج يقرأ ملفا يحتوي على نتائج اختبارات مجموعة من الطلبة وإصدار تقرير له. يحتوي كل رقم الطالب، الاسم، ونتيجة الاختبار (من 50)، وعلى البرنامج أن يحسب النتيجة كنسبة مئوية، ويطبع رقم الطالب، الاسم، الدرجة، والمستوى معبرا عنه بحرف أبجدي.

Design a program which will read a file of student test results and produce a _STUDENT TEST GRADE report. Each record contains student's number, name, and test score (out of 50). The program is to calculate for each student the test score as a percentage and to print the student's number, name, test score (out of 50) and a letter grade on the report. The letter grade is determined as follows:

A = 90-100%
B = 80-89%
C = 70-79%
D = 60-69%
F = 0-59%

A- Defining diagram

Input	Processing	Output
student test record	Print heading	heading line
. student-number	Read student record	student details
. name	Calculate test percentage	student_number
. test_score	Calculate grade	. name
	Print student details	. test_score
		. grade

B- Control structures required

A DOWHILE loop to control the repitition
A linear IF statement to calculate the grade, and
A formula to calculate the percentage.

(لا يمكن استخدام هيكل **CASE** هنا حيث إنه غير مؤهل للتعامل مع مدى من القيم، مثـــلا: من 0-50%)

C- Solution algorithm

```
Print_student_record
        Print "STUDENT TEST GRADES"
        Read student record
        DOWHILE not EOF
                percentage = test_score*2
                IF percentage > 89 THEN
                        grade = A
                ELSE
                        IF percentage > 89 THEN
                                        grade = A
                        ELSE
                                IF percentage > 79 THEN
```

```
                    grade = B
                ELSE
                    IF percentage > 69 THEN
                        grade = C
                    ELSE
                        IF percentage > 59 THEN
                            grade =D
                        ELSE
                            grade = F
                        ENDIF
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
        Print student_number, name test_score, grade
        Read student record
    ENDDO
END
```

مثال 6-8 إصدار فاتورة الغاز

تتعامل شركة غاز مع عملائها بتسعيرة على الوجه التالي: الاستهلاك من 60 مترا مكعبا أو أقل، 2 جنيه إسترليني للمتر المكعب، الاستهلاك أكثر من 60 مترا مكعبا، 1.75 جنيها عن 60 متر مكعب، 1.5 جنيه عما يزيد عن ذلك. وتحتفظ الشركة بسجلات العملاء التي تحتوي على رقم العميل، الاسم، العنوان، والاستهلاك.

يراد تصميم برنامج يقرأ السجل، ويحسب سعر الاستهلاك، ويصدر تقريرا يحتوي على رقم العميل، الاسم، العنوان، الاستهلاك، القيمة المستحقة للشركة لكل عميل، ثم إجمالي عدد العملاء وإجمالي قيمة المستحق.

The Domestic Gas Supply Co. records its customers' gas usage on a CUSTOMER USAGE FILE. Each record on the file contains the customer's number, name, address, and gas usage in cubic meters.

The company bills its customers according to the following rate: if the customer's usage is 60 m^3 or less, a rate of $2.00 is applied; if the usage is above 60 m^3 then the rate of $1.75 is applied for the first 60 m^3 and $1.50 for the remaining usage.
Design an algorithm which will read the customer file, produce a report listing each customer number, name, customer usage, and the amount owing.
At the end of the report, print the total number of customers, and the total amount owing to the company.

A- Defining diagram

Input	Processing	Output
customer record	Print heading	heading line
customer_ number	Read usage record	customer details
name	Calculate amount owing	customer_number
address	Print customer details	name
gas_usage	Compute total customers	address
	Compute total amount	gas_usage
	Print totals	amount_owing
		total_customers
		total_amount

B- Control structures required

1.A DOWHILE loop to control the repitition
2.An IF statement to calculate the amount_owing, and
3.Accumulators for total_customers and total_amount.

C- Solution algorithm

```
Bill_gas_records
    Print "CUSTOMER USAGE REPROT" heading
    Set total_customer to zero
    Set total_amount to zero
    Read customer record
    DOWHILE more records
        IF usage <= 60 THEN
            amount_owing = usage *2.00
        ELSE
```

```
        amount_owing = (60*1.75) + (usage - 60)*1.5
    ENDIF
    Print customer_number,name,address,usgae, mount_owing
    Add amount_owing to total_amount
    Add 1 to total_customers
    Read customer record
    ENDDO
    Print total_customers
    Print total_amount
END
```

وتلاحظ في هذا الخوارزم ما يلي:

- عمليات معالجة أجريت قبل الدخول في الدوارة
- المعالجة للسجلات داخل الدوارة
- عمليات معالجة أجريت بعد الخروج من الدوارة

2-6 تدريبات

في المسائل الآتية مطلوب منك:

- تحليل المسألة عن طريق جدول التحليل
- وضع قائمة بمتطلبات الخوارزم
- وضع الخوارزم بأسلوب اللغة شبه البرمجية
- إجراء الاختبار المكتبي لصحة الخوارزم

1- يحوي سجل أصناف مخزنية ما يلي:

- كود السجل (رقم 11)
- رقم الصنف (حرفين وأربعة أرقام)
- اسم الصنف
- الرصيد

مطلوب تصميم برنامج يقرأ هذا الملف ويخرج كافة السجلات المتوافقة مع الكود المذكور والتي يبلغ الرصيد بها صفرا.

A parts inventory record contains the following fields:

◊ record code (only code 11 is valid)
◊ part number (2 alpha and 4 numeric, e.g. AA1234)
◊ part description, and
◊ inventory balance

Design a program which will read this file and print the contents of all the valid inventory records that have zero balance.

2- في نفس المسألة السابقة، يراد تصميم برنامج يخرج تفاصيل كل السجلات المتوافقة، والتي تقع بين القيمتين المبينتين في السؤال، وكذا عدد السجلات المختارة.

Design a program which will read the same file, and prints the details of all valid records whose part numbers fall within the values AA3000 and AA3999 and the count of the selected records.

3- صمم برنامجا يقرأ مصفوفة من 200 حرف ويظهر عدد الحروف الحركية فيها.

Design a program that reads an array of 200 characters and display a count of the vowels (a, e, I, o, u).

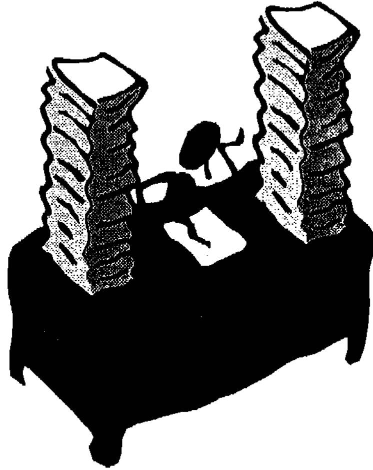
4- شركة كهرباء تمسك سجلات عن استهلاك عملائها في ملف يحتوي على سجلات تفصيلية يحتوي كل سجل على رقم العميل واسمه واستهلاكه خلال الشهر. صمم برنامجا يقرأ الملف المذكور ويخرج تقريرا يبين حساب العملاء بواقع 11 سنتا للكيلووات ساعة حتى 200 ساعة و8 سنت للاستهلاك فوق ذلك ومجموع استهلاك العملاء.

An Electricity Supply Authority records its customers' usage on a file which contains the customer's number, name and usage. Design a program that reads this file and computes the cost of usage at a rate of 11 cent for usage up to 200 KWH and 8 for usage above that. The program will produce a report that contains the details of the customer and the amount due, and the total usage during the month.

5- ملف موظفين يحوى على الاسم والجنس والسن والحالة الاجتماعية، يراد ÷خراج تقريـــــر بعدد المتزوجين وغير المتزوجين والمتزوجات وغير المتزوجات، وأسماء غير المتزوجين من الرجال الذين يزيد عمرهم عن 30 عاما.

a file records the name, gender (M or F) age and marital status (single or married) of employees. Design a program that reads this file and counts the number of married men, single men, married women and single women, and the names of single men above 30 years.

الفصل السابع التحليل البنائي للبرامج



الأهداف:

- التعريف بعملية التحليل البنائي modularization للبرامج
- عرض الخرائط الهيكلية لتصوير الهيكل العام للبرنامج
- إعطاء أمثلة توضيحية لعملية التحليل البنائي للبرامج

الخطوط العامة :

7-1 مفهوم التصميم البنائي البرامج

7-2 الخرائط الهيكلية

7-3 خطوات التحليل البنائي

7-4 أمثلة على التحليل البنائي

7-5 ملخص الفصل

7-6 تدريبات

7-1 مفهوم التحليل البنائي للبرامج

كانت البرامج التي تم عرضها للآن بسيطة، لا تستغرق سوى صفحة واحدة أو جزء من الصفحة في طولها. وكلما زاد تعقد البرنامج تعذرت عملية تصور حل شامل له. في هذه الحالة يصبح من الضروري تقسيم البرنامج إلى مهامه الجزئية، ومعاملة كل مهمة كوحدة مستقلة، هذه الوحدة يطلق عليها "**وحدة** بنائية **module**". ويمكن أن تقسم وحدة بنائية معينة إلى وحدات أقل، وهكذا. وتسمى عملية التجزئة هذه "التحليل الوظيفي **functional analysis**" أو "التصميم من أعلى إلى أسفل **Top-down design**".

وتمكنك عملية تجزئة البرنامج إلى وحدات بنائية من التركيز على كل مهمة على استقلال، ويمكن بعد ذلك استخدام الوحدات البنائية في مشروعات أخرى بنفس صورتها أو بتعديل بسيط.

عملية التجزئة

لإجراء عملية التحليل الوظيفي للبرنامج، اكتب كافة المهام المطلوبة منه كالمعتاد، ثم انظر في تجميع المهام التي يجمعها منطقيا هدف واحد أو وظيفة واحدة. مثل هذه الوظيفة [تسمى "دالة **function**"، أو "برنامجا فرعيا **subroutine**"، أو "إجراء **procedure**"، بحسب اللغة البرمجية المستخدمة] تمثل الوحدة البنائية المستقلة، وتشتمل على عدة أنشطة تهدف جميعها لتحقيق الوظيفة المنشودة. ولا يوجد اصطلاح موحد لمثل هذه الوحدات.

ومن المهم أن تعطى الوحدة البنائية اسما ذا مغزى يدل على مهمتها. وسوف نتبع هنا نفس التقليد؛ فعل أمر يتبعه اسمان.

الدالة الأساسية للبرنامج

طالما تجزأ البرنامج إلى وحدات، فإننا نحتاج إلى وحدة أساسية، أو دالة رئيسية **mainline** تنظم علاقات هذه الوحدات، وتمثل *الخط العام* أو *المنطق العام* في تنفيذ البرنامج. هذا الخط العام هو الذي ينسق بين نشاطات الوحدات المختلفة، فهو الذي يستدعي كل دالة حينما يحين دورها في

التنفيذ. وعملية الاستدعاء هذه تتم بمجرد ذكر اسم الدالة المطلوبة في سطر من أسطر البرنامج الأساسي.

مثال 7-1 قراءة ثلاثة أحرف

سوف نعرض لأسلوب تنفيذ البرنامج السابق ذكره في المثال 4-1 بواسطة التحليل لوحدات بنائية. وسوف نعدل البرنامج بحيث يظل مستمرا إلى أن يدخل المستخدم ثلاثة أحرف هي XXX، الأمر الذي يستلزم إضافة دوارة للبرنامج شرط استمراريتها ألا يكون الحروف الثلاثة معا هي الحرف المذكور.

Design a program which will prompt the user for three characters, accept those characters as inputs, sort them into ascending sequence, and output them on the screen. The program is to continue to accept characters until 'XXX' is entered.

A- defining diagram

Input	Processing	Output
char_1	Prompt for characters	char_1
char_2	Accept three characters	char_2
char_3	Sort three characters	char_3
	Output three characters	

وإليك الخوارزم دون تجزئة:

B- Solution algorithm

Read_three_characters
Prompt the user for char_1, char_2, char_3
Get char_1, char_2, char_3
DOWHILE NOT (char_1=X AND char_2 = X AND char_3 = X)
IF char_1 > char_2 THEN
temp = char_1
char_1 = char_2
char_2 = temp
ENDIF
IF char_2 > char_3 THEN

```
temp = char_2
char_1 = char_3
char_3= temp
ENDIF
IF char_1 > char_2 THEN
temp = char_1
char_1 = char_2
char_2 = temp
ENDIF
Output char_1, char_2, char_3
Prompt the user for char_1, char_2, char_3
Get char_1, char_2, char_3
ENDDO
END
```

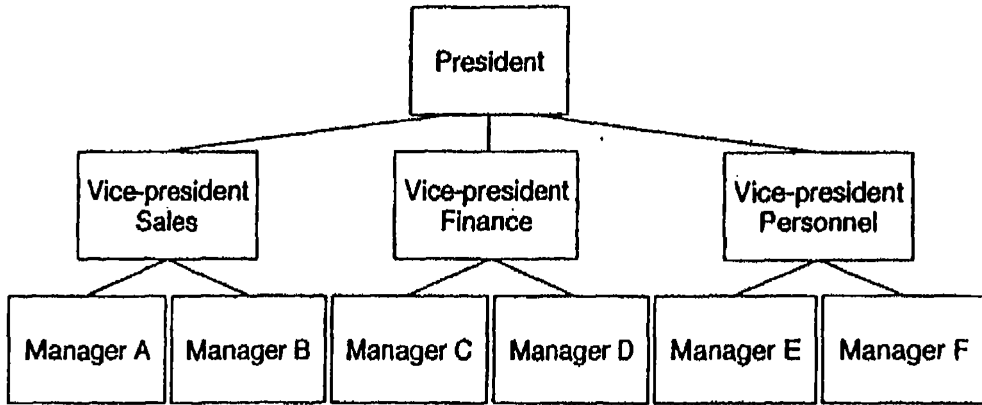
فإذا ما أردنا تجزئة البرنامج لوحدات مستقلة، يتلاحظ أننا يمكن أن يخرج الجزء الخاص بالترتيب من الخوارزم الأصلي، لتكون مهمة قائمة بذاتها، عندئذ يأخذ البرنامج الصورة التالية:

```
Read_three_characters  البرنامج الأصلي (الدالة الأصلية)
Prompt the user for char_1, char_2, char_3
Get char_1, char_2, char_3
DOWHILE NOT (char_1=X AND char_2 = X AND char_3 = X)
 Sort_three_characters  استدعاء دالة الترتيب
 Output char_1, char_2, char_3
 Prompt the user for char_1, char_2, char_3
 Get char_1, char_2, char_3
ENDDO
END
```

7-2 الخرائط الهيكلية

بعد تحليل البرنامج إلى وحداته البنائية، يمكن وضع هذا التحليل في صورة تصويرية عن طريق خرائط تبين التسلسل الهرمي للوحدات، بالضبط كما تصور الهياكل الإدارية في المنظمات، فيتفرع البرنامج الأساسي إلى وحدات بنائية، قد تتفرع بدورها إلى وحدات أصغر، وهكذا.

وقد كان المثال السابق غاية في البساطة، فالبرنامج الأصلي يتفرع عنه دالة واحدة فقط، هي دالة ترتيب الحروف. وبالتالي يمكن تصويره على الوجه التالي:



والآن، إلى مثال أكثر تطورا.

مثال 7-2 حساب أجر عامل

فلنعالج البرنامج المعطى في مثال 4-3 الخاص بحساب أجر العامل، بأسلوب التجزئة. وكما فعلنا في المثال السابق، سوف نجعل البرنامج مستمرا إلى أن ينتهي من قراءة كافة سجلات الملف.

A program is required to <u>read</u> an employee's number, pay rate, and the number of hours worked in a week. The program is then to <u>compute</u> the employee's weekly pay and print it along with the input data. The program is to continue <u>reading</u> employee details until there is no more records on the file.

According to the company's rules, no employee may work more than 60 hours per week, and the maximum hourly rate is $25.00 per hour. If more than 35 hours are worked, then payment for the overtime hours worked is calculated at time-and-a-half. If the hours worked

field or the hourly rate is out of range, then the input data and an error message are printed without calculation made.

A- Defining diagram

Input	Processing	Output
emp_no	Read employee details	emp_no
pay_rate	Validate input fields	pay_rate
hrs_worked	Calculate employee pay	hrs_worked
	Print employee details	emp_pay
		error_mssage

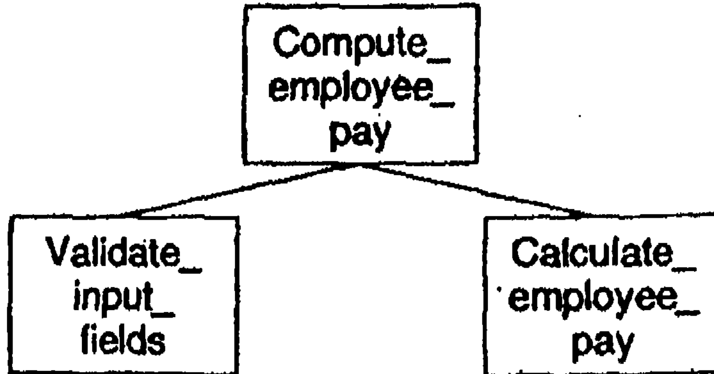
وضع الخوارزم:

B- Solution algorithm

```
Compute_employee_pay
      Read emp_no, pay_rate, hrs_worked
      DOWHILE more records
         Set valid_input to true
         Set error_message to blank
         IF pay_rate > 25.00 THEN
            error_message = "Pay rate exceeds $25.00"
            valid_input = false
            Print emp_no, pay_rate, hrs_worked, error_message
         ENDIF
         IF hrs_worked > 60 THEN
            error_message = "Hours worked exceeds limit of 60"
            valid_input = false
            Print emp_no, pay_rate, hrs_worked, error_message
         ENDIF
         IF valid_input THEN
            IF hrs_worked <=35 THEN
               emp_pay = pay_rate*hrs_worked
            ELSE
               overtime_hrs = hrs_worked - 35
               overtime_pay = overtime_hrs*pay_rate*1.5
               emp_pay = pay_rate*35 + overtime_pay
            ENDIF
```

```
            Print emp_no, pay_rate, hrs_worked, emp_pay
         ENDIF
         Read emp_no, pay_rate, hrs_worked
      ENDDO
END
```

وسوف نعالج هذا الخوارزم المعقد بأسلوب التحليل، حيث تقسمه إلى ثـــلاث دوال، الدالـــة الأصلية، ودالة خاصة بالتحقق من الشروط، ودالة لحساب الأجر، على الوجه التالي:



B- Solution algorithm

```
Compute_employee_pay
      Set valid_input to true
      Read emp_no, pay_rate, hrs_worked
      DOWHILE more records
         validate_input_fieldاستدعاء دالة التحقق
         IF valid_input THEN
            Calculate_employee_payاستدعاء دالة حساب الأجر
            Print emp_no, pay_rate, hrs_worked, employee_pay
         ELSE
            set valid_input to true
         ENDIF
         Read emp_no, pay_rate, hrs_worked
      ENDDO
END
```

```
validate_input_fields
    Set error_message to blank
    IF pay_rate > 25.00 THEN
        error_message = "Pay rate exceeds $25.00"
        valid_input = false
        Print emp_no, pay_rate, hrs_worked, error_message
    ENDIF
    IF hrs_worked > 60 THEN
        error_message = "Hours worked exceeds limit of 60"
        valid_input = false
        Print emp_no, pay_rate, hrs_worked, error_message
    ENDIF
END

Calculate_employee_pay
    IF hrs_worked <=35 THEN
        emp_pay = pay_rate*hrs_worked
    ELSE
        overtime_hrs = hrs_worked - 35
        overtime_pay = overtime_hrs*pay_rate*1.5
        emp_pay = pay_rate*35 + overtime_pay
        Print emp_no, pay_rate, hrs_worked, emp_pay
    ENDIF
END
```

وكما ترى، يتم تبادل التحكم في تنفيذ البرنامج بين الدوال الثلاث، تحت سيطرة الدالة الأساسية، فينقل التنفيذ إلى إحدى الدوال حين تستدعيها الدالة الأساسية، وبعد أن تنهي الدالة الفرعية عملها وتعيد التحكم في البرنامج للدالة الأساسية.

3-7 خطوات التحليل البنائي

عملية تجزئة البرنامج يمكن أن تكون سهلة إذا التزمت باتباع الخطوات التالية:

1. حدد المشكلة عن طريق تحليلها إلى العناصر الأساسية، وهو المدخلات والمعالجة

والمخرجات. ويجب أن تضم المعالجة كافة الأنشطة المطلوبة من البرنامج.

2. جمع الأنشطة في مجموعات منطقية من الناحية العملية. ركز في هذه الخطوة على الوحدات الخاصة بالمستوى الأول من التحليل، ولاحظ أنه ليس من الضروري أن تستغرق هذه الخطوة كل الأنشطة، فالبعض منها قد يتعرض له عند مستوى أدنى من التحليل.
3. اصنع خريطة التدرج الهيكلي للوحدات البنائية، ومنه يتضح دور كل دالة في التنفيذ.
4. ضع خوارزم الدالة الأساسية والذي يتضح منه المنطق العام للبرنامج وتسلسل استدعاءات الدوال الفرعية.
5. صغ خوارزم كل دالة فرعية
6. قم بإجراء الاختبارات المكتبية على الدالة الرئيسية والدوال الفرعية.

7-4 أمثلة على التحليل البنائي

سوف نتبع في الأمثلة التالية الخطوات الستة التي عرضنا لها في القسم السابق.

مثال 7-4 إصدار تقرير طلبيات

تريد شركة أن تصدر تقريرا بالطلبيات التي تتلقاها، وتحتفظ ببياناتها في ملف لهذا الغرض. ويحتوي كل سجل في هذا الملف على رقم الصنف المطلوب، اسم الصنف، الكمية المطلوبة، سعر الوحدة، سعر النقل للوحدة، وسعر التغليف للوحدة. ويكون التقرير على الصورة المبينة، ويحتوي على اسم للتقرير، وأسطر تضم رقم الصنف، اسم الصنف، الكمية المطلوبة، والسعر الإجمالي للطلبية. ويجب ألا تزيد الصفحة عن 45 سطرا.

ويحسب خصم بعد حساب ثمن الأصناف مقداره 10% للطلبيات التي تزيد قيمتها عن مائة جنيه، ثم يضاف بعد ذلك سعر الشحن والتغليف. والمراد وضع البرنامج الذي يصدر هذا التقرير.

The Acme Spare Parts Co. wants to produce an Order Report from its Product Order File. Each record on the file contains the product number of the item ordered, the product description, the number of

units ordered, the retail price per unit, the freight charges per unit, and the packaging cost per unit.
The output report is to contain heading and column heading as specified in the following chart:

	ACME SPARE PARTS ORDER REPORT		PAGE NO
PRODUCT NO	PRODUCT DESCRIPTION	UNITS ORDERED	TOTAL AMOUNT DUE
xxxxxxx	xxxxxxxx	xxxxxxxxx	xxxxxxxxx
xxxxxxx	xxxxxxxx	xxxxxxxxx	xxxxxxxxx

There must be an allowance of 45 lines per page.
A discount of 10% is allowed on the amount due for all orders over $100.00. The freight charges and packaging costs per unit must be added to the resulting value to determine the total amount due.

أولا: وضع الجدول التحليلي:

A- Defining diagram

Input	Processing	Output
Order record	Print heading	main heading
. prod_number	Read order records	column headings
. prod_descrp	Calculate amount due	page number
. no_of_units	Calculate discount	detail lines
.	Calculate freight charge	. prod_number
.	Calculate packaging	. prod_descrp
.	Print order details	. no_of_units
.	Compute page increments	total_amount_due

ثانيا: تجميع الوظائف في وحدات بنائية

B- Group the activities into modules

1- الوظيفة الأولى التي سوف يتم عزلها كوحدة مستقلة هي طباعة العنـوان، والـتي عرفـت بالعبارة:

'Print heading"

وسوف نسمي هذه الوظيفة: print_page_heading

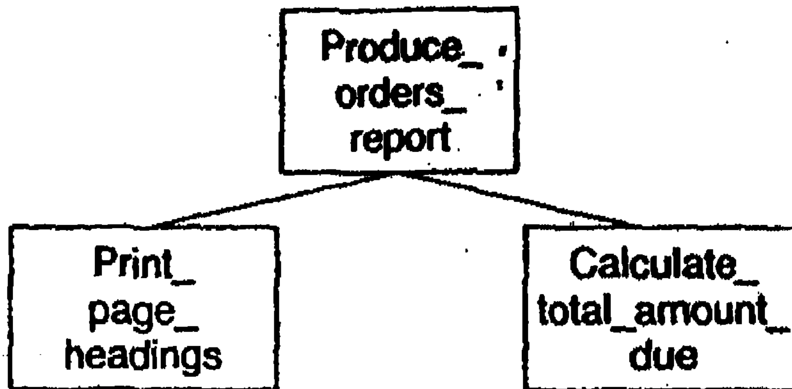
2- أما الوظيفة الثانية فهي المتعلقة بإجراء الحسابات، وتجمع الأنشطة التالية:

Calculate amount due
Calculate discount
Calculate freight charge
Calculate packaging

وهي وظيفة ذات هدف واحـــد هـــو اســتخراج القيمـــة الإجمالــة للطلبيـــة ونســميها:

Calculate_total_amount

ثالثا: تكوين الخريطة الهيكلية وتكون على الصورة التالية:



وهي التي تصور الهيكل العام للخوارزم، وتتكون من دالة أساسية وبرنامجين فرعيين.

رابعا: وضع خوارزم الدالة الأساسية

يحتاج الخوارزم الخاص بالدالة الأساسية ما يلي:

1. دوارة لإجراء عمليات التكرار؛ من نوع DOWHILE
2. استدعائين للدالتين
3. عداد لدالة طباعة العنوان لتخزين أرقام الصفحات
4. عداد لاسطر الصفحات

وتكون صياغة الخوارزم كالتالي:

Print_orders_report
Set page_counter to zero استهلال عداد الصفحات بالصفر
Print_page_heading استدعاء دالة طباعة العنوان
Set line_counter to zero استهلال عداد الأسطر بالصفر

```
Read order record
DOWHILE more records
 IF line_counter >= 45 THEN
  print_page_heading
```

استدعاء دالة الطباعة بعد السطر 45

```
  Set line_counter to zero
 ENDIF
Calculate total_amount
```

استدعاء دالة حساب الإجمالي

```
Print prod_number, prod_descrp, no_of_units, total_amount
Add 1 to line_counter
Read order record
ENDDO
END
```

خامسا: ضع خوارزم كل دالة فرعية:

أ) دالة الطباعة:

```
Print_page_heading
 Add 1 to page_counter
 Print main heading 'ACME SPARE PARTS'
 Print heading 'ORDER REPORT'
 Print coulumn heading 1
 Print coulumn heading 2
 Print blank line
END
```

ب) دالة حساب القيمة الإجمالية:

```
Calculate_total_amount
 amount_due = no_of_units*retail_price
 IF amount_due > $100 THEN
  discount = amount_due*0.1
 ENDIF
 amount_due = amount_due - discount
 freight_due = freight_charge* no_of_units
 packaging_due = packaging_cost* no_of_units
```

total_amount = amount_due + freight_due + packaging_due
END

سادسا: اختبر الحل مكتبيا

لا يختلف الاختبار المكتبي للبرنامج المجزء عنه في البرنامج غيرالمجزء. ولما كانت الدالتان الفرعيتان لا تزيدان عن عدة خطوات متتابعة، فلا داعي أن يختبر أي منهما على استقلال، ولذا فسيجري اختبارهما مع اختبار الدالة الأصلية.

i- Input data

سوف نأخذ ثلاثة مجموعات من المدخلات الاختبارية.

Record	no_of_units	retail_price	freight_charge	packing_cost
1	10	1.00	0.20	0.50
2	20	2.00	0.10	0.20
3	100	3.00	0.10	0.20
EOF				

ii- Expected results

	ACME SPARE PARTS ORDER REPORT		PAGE NO.
PRODUCT NO	PRODUCT DESCRIPTION	UNITS ORDERED	TOTAL AMOUNT DUE
100	xxxxxxxx	10	17.00
200	xxxxxxxx	20	46.00
300	xxxxxxxx	100	300

iii- Disk checking table

Statement	DOWHILE OK?	page_counter	line_counter		no_of_units	retail_price	freight_charge	packg_cost	total_amount
Initialize		0							
Print_page_heading		1							
Initialize			0						
Read				100	10	1.00	0.20	0.50	
DOWHILE	yes								
IF									
Calculate									17.00
Print					print				print
Add			1						
Read				200	20	2.00	0.10	0.20	
DOWHILE	yes								
IF									

Statement	DOWHILE OK?	page_ counter	line_ counter		no_of_ units	retail_ price	freight_ charge	packg_ cost	total_ amount
Calculate									46.00
Print					print				print
Add			2						
Read				300	100	3.00	0.10	0.20	
DOWHILE	yes								
IF									
Calculate									300.00
Print					print				print
Add			3						
Read				EOF					
DOWHILE	no								
END									

وتلاحظ في هذا الاختبار:

جعلنا عدد الأصناف ثلاثة فقط للتبسيط.

لم نختبر دالة الطباعة لعدم وصول عدد الأسطر للحد الأقصى، ويمكن تنفيذ ذلك بخفض العدد الأقصى للأسطر مؤقتا لسطرين مثلا، فيطبع عنوان التقرير عند الانتقال للصفحة التالي لكتابة الصنف الثالث.

لم نضف اسم الصنف للقائمة لعدم تأثيره على مجرى التنفيذ، فكل ما في الأمر أنه يطبع مع بقية المخرجات المطبوعة.

مثال 7-5 حساب تكاليف ترخيص مركبة

يراد وضع برنامج يحسب تكاليف ترخيص مركبة جديدة.

والبرنامج تفاعلي، بمعنى أنه يتقبل البيانات من وحدة إدخال ويظهر المخرجات على الشاشة.

والبيانات المدخلة هي:

- اسم صاحب السيارة
- صانع السيارة
- الموديل
- الوزن بالكيلوجرام
- نوع الشاسيه (سيدان أو صالون)

- خاصة أم تابعة لمؤسسة
- السعر

ويضاف لمصاريف الترخيص ضريبة عامة بواقع جنيهان عن كل مائة جنيــــه مــن الســعر أو كسورها.

وتحسب مصاريف الترخيص كالتالي:

رسوم ترخيص27 جنيه

ضريبة محلية خاصة 5%تابعة لمؤسسة 7%

ضريبة على الوزن خاصة 1% تابعة لمؤسسة 3%

قسط تأمين خاصة 1% تابعة لمؤسسة 2%

وتضم بيانات البرنامج المخرجة ما يلي:

- صانع السيارة
- الموديل
- نوع الشاسيه
- مصاريف الترخيص
- ضريبة محلية
- ضريبة الوزن
- قسط التأمين
- جملة مصاريف الترخيص
- ضريبة عامة
- القيمة الإجمالية (إجمالي مصاريف الترخيص + الضريبة العامة)

ويظل البرنامج يتلقى البيانات حتى يدخل المستخدم الحروف XXX في بيان الاسم.

A program is requited to calculate and print the registration cost of a new vehicle.

The program is to be interactive, that is, all the inputs will be provided at a terminal on the salesperson's desk. The program will

then calculate the related costs and return the information to the screen.
The input details required are:
Owner's name
Vehicle make
Vehicle model
Weight (in kg)
Body type (sedan or wagon)
Private or business code ("P" of "B")
Price
A federal tax is also to be paid. This is calculated at the rate of $2,00 for each $100.00 of the price, or part thereof.
The vehicle registration cost is calculated as the sum of the following charges:
Registration fee $27.00
Tax levy PRIVATE 5% BUSINESS 7%
Weight tax 1% 3%
Insurance premium 1% 2%
The program is to calculate the total registration charges plus the federal tax, and to print the following information:
Vehicle make
Vehicle model
Body type
registration fee
Tax levy
Weight tax
Insurance premium
Federal tax
Total amount payable
(Total amount payable = total registration charges + federal tax)
The program is to process costs until an owner's name of 'XXX' is entered.

أولا: وضع الجدول التحليلي

A- Defining diagram

Input	Processing	Output
owner_name	Get input details	vehicle_make
vehicle_make	Calculate tax_levy	vehicle_model
vehicle_model	Calculate weight tax	body_type
body_type	Calculate ins_premium	reg_fee
weight	Calculate total_reg_cost	tax_levy
usage_code	Calculate federal_tax	weight_tax
price	Calculate total_amount	ins_premium
		total_reg_cost
		federal_tax
		total_amount

ثانيا: تجميع الوظائف في وحدات بنائية

سوف نجمع العمليات المبينة في ثلاثة وظائف أساسية:

1- إدخال البيانات وتسمى Get_vehicle_data

2- إظهار البيانات وتسمى Display_registration_data

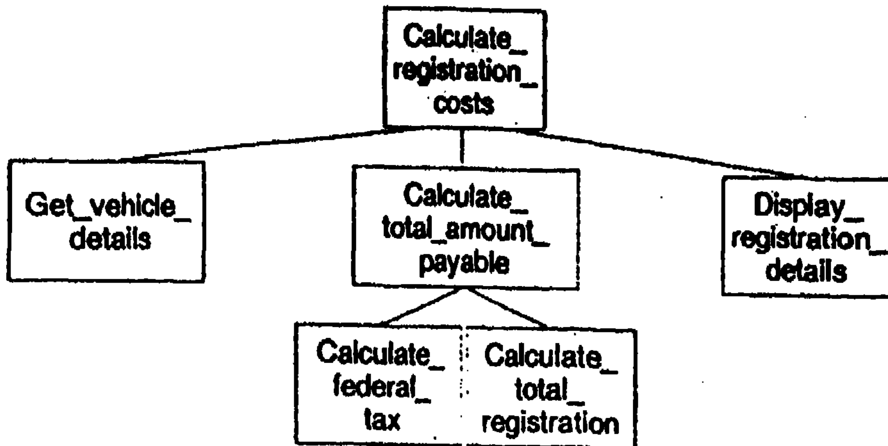
3- عمليات الحساب وتجمع في وظيفة واحدة تسمى Calculate_total_amount

هذه العملية سوف تجزأ إلى وظيفتين:

1-3 حساب الضريبة العامة، وتسمى Calculate_federal_tax

2-3 حساب مصاريف التسجيل، وتسمى Calculate_total_registration

ثالثا: تكوين الخريطة الهيكلية وتكون على الصورة التالية:



رابعا: وضع خوارزم الدالة الرئيسية:

يطلب خوارزم الدالة الرئيسية ما يلي:

- دوارة DOWHILE للتحكم في التكرار
- استدعاء الدوال بحسب الحاجة

```
Calculate_registration_cost
 Read owner_name
 DOWHILE owner_name NOT 'XXX'
  Get_ vehicle_data
  Calculate_total_amount
  Display_registration_data
  Read owner_name
 ENDDO
END
```

ويظهر من ذلك مدى البساطة في البرنامج الأصلي، وهو ميزة حقيقية لأسلوب التجزئة للبرامج، فهو يقتصر على عمل الدوارة واستدعاء الدوال التي تقوم بالوظائف.

خامسا: وضع خوارزم الدوال الفرعية

1- دالة إدخال البيانات: تقوم هذه الدالة بتقبل البيانات المدخلة من المستخدم:

```
Get_vehicle_data
 Get vehicle_make
 Get vehicle_model
```

```
 Get body_type
 Get usage_code
 Get price
END
```

2- دالة الحسابات

ويقتصر خوارزمها على استدعاء الدالتين الفرعيتين التابعتين لها:

```
Calculate_total_amount
 Calculate_federal_tax
 Calculate_total_registration
 total_amount = federal_tax + total_reg_cost
END
```

2-1 دالة حساب الضريبة العامة

2-1 دالة حساب الضريبة العامة

تتطلب هذه الدالة تقسيم السعر فئات قيمة كل فئة 100 جنيه، ونستخدم لذلك متغيرا هو tax_unit لكي يعد عدد أقسام السعر المساوية لـ 100 جنيه، ونوالي طرح قيمة 100 من السعر، وفي كل مرة يكون الطرح ممكنا (السعر أعلى من 100) يتزايد العداد بمقدار الوحدة. ثم تكون معادلة حساب الضريبة هي ضرب القيمة التي وصل إليها العداد، مضافا إليها 1 (حيث أن الباقي سوف يدخل في الحساب أيضا)، في 2.

```
Calculate_federal_tax
 set tax_unit - zero
 DOWHILE price > $100.00
  price = price - 100
  increment tax_unit by 1
 ENDDO
 federal_tax = (tax_unit +1)*2
END
```

2-2 دالة حساب مصاريف التسجيل:

```
Calculate_ total_registration
 registration_fee = 27
 IF usage_code = 'P' THEN
```

```
tax_levy = price*0.05
weight_tax = price*0.01
ins_premium = price*0.01
ELSE
tax_levy = price*0.07
weight_tax = price*0.03
ins_premium = price*0.02
ENDIF
total_ reg_cost = registration_fee + tax_levy + weight_tax +
ins_premium
END
```

3- دالة إخراج البيانات

```
Display_registration_data
 Display vehicle_make
 Display vehicle_model
 Display body_type
 Display reg_fee
 Display tax_levy
 Display weight_tax
 Display ins_premium
 Display total_reg_cost
 Display federal_tax
 Display total_amount
END
```

سادسا: الاختبار المكتبي

بما أن البرنامج يتفرع إلى اتجاهين، فإن مجموعتين من المدخلات الاختبارية يكونان كافيين.

i- input data

record	weight	usage_code	price
1	1000	P	30.000
2	2000	B	20.000
XXX			

ii- Expected results

	Record1	Record2
reg_fee	27.00	27.00
tax_levy	1500.00	1400.00
weight_tax	10.00	60.00
ins_premium	300.00	400.00
total_reg_cost	1837.00	1887.00
federal_tax	600.00	400.00
total_amount	2437.00	2287.00

iii- Check desk table

Statement	owner's name	DOWHILE OK?	weight	usage code	price	federal tax	total_reg_cost	total_amount
Initialize								0
Read	1							
DOWHILE		yes						
Get			1000	P	30.000			
Calculate						600	1837	2437
Display						yes	yes	yes
Read	2							
DOWHILE		yes						
Get			2000	B	20000			
Calculate						400	1887	2287
Display						yes	yes	yes
Read	XXX							
DOWHILE		no						
END								

ملخص الفصل

قدما في هذا الفصل أسلوبا بنائيا لتصميم البرامج، عرفنا به مفهوم التصميم من أعلى إلى أسفل ومفهوم الوحدات البنائية. كما قدمنا أمثلة توضح مزايا هذا الأسلوب.

كما عرضنا للخرائط الهيكلية كوسيلة لتصوير الهيكل العام للبرنامج، حيث تتدرج الوحدات البنائية في شكل تدرج هرمي.

وتضمن الفصل الخطوات الواجب اتباعها للحصول على تصميم جيد للبرامج: تحليل المسألة، تجميع الأنشطة في وظائف محددة، تكوين الخريطة الهيكلية، وضع خوارزم الدالة الرئيسية الذي

يحدد المنطق العام للبرنامج، وضع خوارزمات الدوال الفرعية، ثم الاختيـــار المكتـــبي للـــدوال. وأعطيت أمثلة على تطبيق هذه الخطوات.

تدريبات

اتبع خطوات التصميم البنائي المبينة في الفصل لوضع خوارزم البرامج التالية:

1- اكتب خوارزما لبرنامج يقرأ عمليات العملاء البنكيين، بحيث تكون بيانات العمليات هـي: الرصيد الحالي، مبلغ العملية، كود العملية (سحب أو إيداع)، الرصيد الجديد. يخرج البرنـــامج الرصيد الجاري والرصيد الجديد ومبلغ العملية، وتحذيرا عند عدم كفاية الرصيد. عنـــد انتـــهاء العمليات يدخل المستخدم حرف Q.

Design a algorithm which receives current balance and the amount of the financial transaction and its code (D for deposit, C for check). A new balance is to be calculated. The program is to display the initial balance, amount of transaction, new balance, and a warning message if the balance is negative. Letter Q is entered at the end of data.

2- تريد شركة منح عمالها زيادة بمقدار 8%، ولكنها تريد أولا أن تعرف ما تكلفـــه هـــذه العلاوة. ضع برنامجا يستحث ويقرأ اسم العامل، أجره الحالي، أجره بعد الزيادة، وإجمالي الأجور قبل الزيادة وبعدها ومقدار الزيادة.

A company is considering a pay increase of 8%, but before doing so it want to know how much this increase will cost. Design a program that will prompt for the name and salary of every employee, compute the increase and displays the salary before and after the increase, total payroll before and after increase, and the amount of increase.

3- شركة تليفونات تحتفظ بملف لحسابات عملائها يحتوي على اسم العميل ورقم هاتفه ورقـــم الهاتف المكالمة المطلوبة ومدة المكالمة وبعد مسافتها. المطلوب وضع برنامج يقرأ هذه البيانـــات ويخرج تقريرا على الشكل المبين في السؤال.

A telephone company's charge file contains the subscriber's name, phone number, distance and duration of the call. Design a program

that reads the file, calculates the charges, and prints the following report:

	TIDY PHONE TELEPHONE CHARGES		PAGE XX
SUBSRIB ER NAME	SUBSCRIB ER NUMBER	PHONE NUMBER CALLES	COST OF CALL
XXXX	XXXXXX	XXXXX	999.99
XXXXX	XXXXX	XXXXX	999.99
		TOTAL	999.99

The cost is calculated as follows:

Distance	Cost ($/Minute)
less than 25 Km	0.35
25 -74	0.65
75-299	1.00
300-999	2.00
1000 and more	3.00

4- تريد شركة توزيع منتجات أن تضع برنامجا لمعالجة طبيات البيع، يتقبل بيانات الطلبية وينتج قسيمة الشحن. المطلوب وضع برنامج تفاعلي يتقبل البيانات المدخلة طبقا للنظام المبين في السؤال. يصدر البرنامج رسالة خطأ إذا لم يكن رقم الصنف المدخل خمسة أرقام. إذا لم يكن الرصيد كافيا يكتب سطر الصنف المطلوب مبينا به الكمية الموجودة بالمخزن مع رسالة تفيد عدم الوفاء بالطلبية كلها.

A company requires a program that accepts order details and generates Shipping List. Design an interactive program with the following dialogue screen:

Enter Item no.: 99999 (خمسة أرقام) رقم الصنف

Quantity on Hand:999 (ثلاثة أرقام) الكمية الموجودة

Enter quantity: 999 (ثلاثة أرقام) الكمية المطلوبة

Order number: 999999 (ستة أرقام) رقم الطلبية

If the item number is not five digits an error message is to be printed.

If the quantity on hand is not sufficient, s message "Order Partially Filled" is printed on the REMARK field of the Shipping List..
If the quantity on hand is zero, the message issued is "Out of Stock".
The program tuns till a number 0 is entered in Item no.
The report layout is as follows:

	MTRE -11 HARDWARE SHIPPING LIST		PAGE XX
ORDER NO	ITEM NO	UNITS SHIPPED	REMARKS
999999	99999	999	-----
999999	99999	999	-----

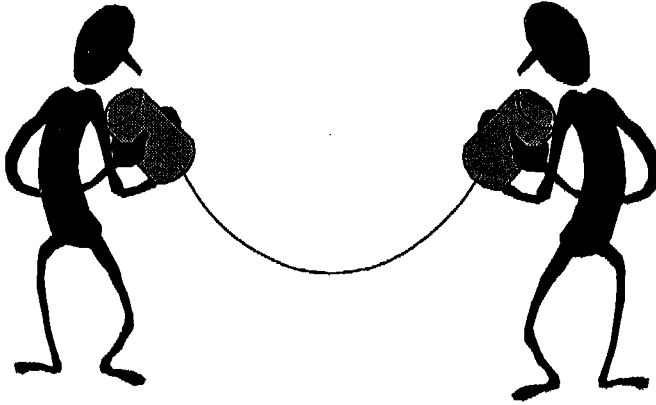
5- في السؤال السابق، تريد الشركة تطوير البرنامج ليخرج وثيقة لمراقبة المخزون، تبين الأصناف التي طلبت، والرصيد المتبقي بعد تنفيذ الطلبية. بالنسبة للأصناف التي لم يمكن تحقيق طلبها لنفاد المخزون، تطبع الكمية التي كانت مطلوبة.

The program of the last problem is to be modified to produce a Back Order List of the following layout:

	MTRE -11 HARDWARE BACK ORDER LIST	PAGE XX
ORDER NO	ITEM NO	BACK ORDER QUY
999999	99999	999

If the quantity on hand is zero, the full quantity ordered is to be printed on this list.

الفصل الثامن المعاملات والوحدات البيانية



الأهداف:

- التعريف بالمتغيرات وأنواعها
- التفرقة بين الأنواع الأولية من البيانات وهياكل البيانات
- مناقشة الاتصال بين الدوال وإمرار المعاملات بينها
- التعريف بأسلوب التصميم الكائني والمصطلحات المتصلة به

الخطوط العامة :

18-1 بيانات البرنامج

8-2 الاتصال بين الوحدات البيانية

8-3 استخدام المعاملات في تصميم البرامج

8-4 التصميم الكائني

8-5 ملخص الفصل

8-6 تدريبات

1-8 بيانات البرنامج

ركزنا في الفصول السابقة على تصميم خوارزم البرامج. ولكن هذه البرامج إنما يتم وضعها لكي تعالج بيانات معينة، ولذا فلا بد للمبرمج أن يكون فكرة واضحة لطبيعة ببيانات البرامج وأنواعها المختلفة. فقد تكون البيانات متغيرات منفردة، كمتغيرات الأعداد الصحيحة أو الحرفية، وقد تتجمع في هياكل كالمصفوفات أو الملفات.

المتغيرات والثوابت والثوابت الحرفية

"المتغير **variable**" هو اسم يعطى لمجموعة خلايا في الذاكرة تخصص لتخزين بيان معين قيمته معرضة للتغير أثناء عمل البرنامج، فالمتغير **total_amount** مثلا يأخذ قيما مختلفة طبقا لظروف التشغيل.

وعلى عكس ذلك يكون "**الثابت constant**"، فهو بيان له اسم وقيمة، ولكن القيمة تكون ثابتة طوال تنفيذ البرنامج [من ذلك مثلا النسبة التقريبية، فهي تعطى عادة الاسم "ط"، وقيمتها ثابتة، وهي 3.14 تقريبا]. والثابت الحرفي هو بيان يأخذ اسمه نفس قيمته، كأن يكون اسمه '50'.

الأنواع الأولية للبيانات

تنقسم البيانات بصفتها المنفردة إلى أنواع أولية، مثل:

الأعداد الصحيحة integer: وتمثل عددا صحيحا موجبا أو سالبا.

الأعداد الحقيقية real: وتتكون من الأعداد ذات الكسور العشرية.

المحارف characters: وهي الحروف الأبجدية **letters** والعلامات الخاصة مثل علامات الاستفهام أو الفاصلات وغيرها. [يمكن للأعداد أن تعتبر من المحارف إذا لم تكن داخلة في عمليات حسابية، فأرقام الهاتف تعتبر من ناحية البرمجة من المحارف وليست من الأعداد].

المتغيرات البولية Boolean: [وتسمى أيضا المنطقية **logical**] وتكون لها إحدى حالتين، إما صحيحة (متحققة) **true** أو خاطئة (غير متحققة) **false**. هذه المتغيرات تعمل عمل مفتاح

تحويل للبرنامج، فتؤدي إلى تحويل مسار التنفيذ بحسب حالتها. [من أمثلة ذلك الجنس، فهو إما ذكر أو أنثى، راجع الأمثلة الخاصة بالطلبة]

هياكل البيانات

هياكل البيانات هي بيانات مجمعة في وحدات، قد تكون من الأنواع الأولية أو من هياكل ذات مستوى أدنى. وتتشكل البيانات في هياكل تعبر عن الصورة التي يراها عليها البرنامج، وأشهر هياكل البيانات هي:

السجلات records: مجموعة من حقول **fields** يحتوي كل حقل على بيان معين. فسجل الطالب مثلا يحتوي على بيانات اسمه وعنوانه والمواد المسجل بها...الخ.

الملفات files: وهي تجميعات السجلات، كملف للطلبة.

المصفوفات arrays: مجموعات من المتغيرات أو البيانات لها نفس النوعية، ويتعامل معها بنفس الاسم، ويتم التعامل مع كل عنصر من عناصر المصفوفة بواسطة هذا الاسم ودليل يبين موضع العنصر في المصفوفة. ويكون الدليل إما رقما تابعا للمصفوفة، أو على شكل رقم سفلي subscript. فعلى سبيل المثال، قد تنشئ مصفوفة باسم **score** تخزن فيها درجات الطلبة، فيكون درجة الطالب الثالث فيها هي score(3) أو $score_3$.

العبارات النصية strings: وهي مجموعة من المحارف تشكل عبارة واحدة، مثل "Jenny Parker" كاسم لأحد الطلبة.

8-2 الاتصال بين الوحدات البنائية

حينما تفكر في تجزئة البرنامج، ليس المفروض أن تقصر تفكيرك على الوحدات البنائية له، بل يجب أن تفكر في أسلوب سريان البيانات بين تلك الوحدات. ويمكن تحقيق مثل هذا الاتصال عن طريق البيانات العامة، وقوائم المعاملات.

البيانات العامة

في كافة الأمثلة التي تقدمت، كان المتغير يستدعى من دالة إلى أخرى بنفس الاسم. مثل هذه البيانات، والتي يمكن أن تستدعى من أية دالة من دوال البرنامج، تسمى **"بيانات عامة Global data"**. ومع تقدمك في البرمجة ستعلم أنه ليس ضروريا أن تكون كافة المتغيرات شائعة.

البيانات المحلية

البيانات التي تعرف داخل دالة ما تكون مقصورة عليها، ومن ثم تسمى **"بيانات محلية local data"**، هذا القصر له قيمة بالغة الأهمية في منع أخطاء تنفيذ البرامج، فهو يقلل ما يسمى "التأثيرات الجانبية" في التنفيذ.

مجال البيانات

مجال البيانات scope of data هو المنطقة التي يعرف فيها البيان، ويتم التعامل معه فيها. فمجال البيانات العامة هي البرنامج بأكمله، ومجال البيانات المحلية هي الدالة المعرف بها.

التأثيرات الجانبية

يقصد بالتأثيرات الجانبية قيام دالة فرعية بتغيير قيمة متغير في دالة أخرى من دوال البرنامج، وهي صورة من صور الاتصال الداخلي للبرامج. وليس هذا بالأمر الضار في حد ذاته، ولكنه يؤثر سلبا على القدرة على تفهم البرنامج ككل، ومن ثم فإنه يتطلب أكبر قدر من الحرص.

فالبرامج بعد وضعها تتعرض لمتطلبات التعديل والتغيير، وكثيرا ما يكون مطلوبا تعديل برنامج ما، ويقوم بذلك مبرمج خلاف واضع البرنامج، فإذا كان غير ملم تماما بكيفية الاتصال بين الدوال، فإن تعديله قد يؤدي لنتائج خطيرة على البرنامج.

إمرار المعاملات

الطريقة الثانية للتعامل بين الدوال هي إمرار العوامل **parameter passing**. والمعـــامل قـــد يكون أحد المتغيرات، أو الثوابت، أو الثوابت الحرفية التي يمكنها الاتصال بين أجزاء البرنـــامج. وقد تكون هذه الطريقة على إحدى ثلاث صور:

- أن تمرر المعلومة من الدالة المستدعية إلى الدالة المستدعاة، لتقوم الأخـــيرة باســتخدامها في عمليات المعالجة الخاصة بها، وينتهي الأمر عند ذلك.
- أن تمرر المعلومة من الدالة المستدعاة إلى الدالة المستدعية، لتستخدمها هذه الأخيرة.
- أن تمرر المعلومة من الدالة المستدعية إلى الدالة المستدعاة، لتعالجها كما في الحالــة الأولى، ثم تعيد نتيجة المعالجة للدالة المستدعية.

والمعاملات التي تمرر بين الدوال قد تكون معاملات بيانات، أو معاملات حالات، ويمكـــن أن يتضمنها الشكل الهرمي للبرنامج على الصورة التالية:



for status parameters



for data parameters

فمعاملات البيانات تحتوي على المتغير أو البيان الذي ينقل من دالة لأخـــرى، أمـــا معـــاملات الحالات فتعمل كمفتاح تحويل في سير الدالة المنتقل إليها المعامل. ولا يجب اســـتخدام معـــامل بيانات للقيام بهذه المهمة في نفس الوقت، فإن ذلك يؤدي إلى عيبـــين جســـيمين في تصميـــم البرنامج:

- قد يتسبب ذلك في بلبلة القارئ حيث استخدم المتغير لأكثر من مهمة (زيـــادة التحميـــل overloading).
- قد يتسبب ذلك في أخطاء عند تعديل البرامج مستقبلا، إذ لا يكون واضحا قيـــام المتغـــير بمهمتين في نفس الوقت.

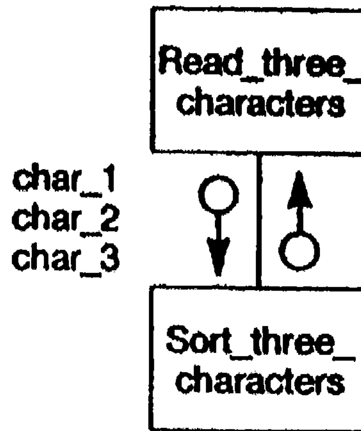
3-8 استخدام المعاملات في تصميم البرامج

لتوضيح استخدام وسيلة إمرار المعاملات بين الدوال، سوف نعرض للمثال 7-1 مرة أخـــرى، لنرى كيف تكون صياغته في هذه الحالة.

مثال 8-1 قراءة ثلاثة أحرف

Design a program which will <u>prompt</u> the user for three characters, <u>accept</u> those characters as inputs, <u>sort</u> them into ascending sequence, and <u>output</u> them on the screen. The program is to continue to accept characters until 'XXX' is entered.

ويمكن تصوير الاتصال بين دالتي البرنامج على الوجه التالي:



تعديل الخوارزم:

تمثل المعاملات الخاصة بالدوال كمتغيرات محتواة بين قوسين يتبعان اسم الدالة، وذلك في صياغة الدالة نفسها وكذا في أمر استدعائها، على الوجه التالي:

```
Read_three_characters
    Prompt the user for char_1, char_2, char_3
    Get char_1, char_2, char_3
    DOWHILE NOT (char_1=X AND char_2 = X AND char_3 =
X)
```

```
Sort_three_characters (char_1, char_2, char_3) لاحظالمعاملات بين القوسين
Output char_1, char_2, char_3
      Prompt the user for char_1, char_2, char_3
      Get char_1, char_2, char_3
   ENDDO
END
```

```
Sort_three_characters (char_1, char_2, char_3)لاحظ المعاملات بين القوسين
   IF char_1 > char_2 THEN
    temp = char_1
    char_1 = char_2
    char_2 = temp
   ENDIF
   IF char_2 > char_3 THEN
    temp = char_2
    char_1 = char_3
    char_3= temp
   ENDIF
    IF char_1 > char_2 THEN
    temp = char_1
    char_1 = char_2
    char_2 = temp
   ENDIF
END
```

ولنأخذ المثال التالي له لنضعه على نفس الصورة.

مثال 8-2 حسابه أجر عامل

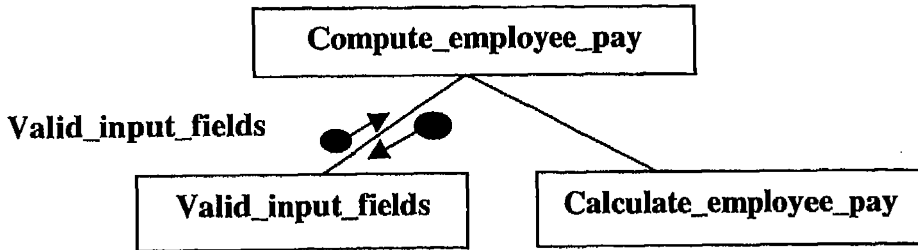
A program is required to <u>read</u> an employee's number, pay rate, and the number of hours worked in a week. The program is then to <u>compute</u> the employee's weekly pay and print it along with the input data. The program is to continue <u>reading</u> employee details until there is no more records on the file.

According to the company's rules, no employee may work for more than 60 hours per week, and the maximum hourly rate is $25.00 per hour. If more than 35 hours are worked, then payment for the overtime hours worked is calculated at time-and-a-half. If the hours worked field or the hourly rate is out of range, then the input data and an error message are printed without calculation made.

A- Defining diagram

Input	Processing	Output
emp_no	Read employee details	emp_no
pay_rate	Validate input fields	pay_rate
hrs_worked	Calculate employee pay	hrs_worked
	Print employee details	emp_pay
		error_mssage

ويكون التصوير لشكل البرنامج كالتالي، ومنه نرى أننا سوف نركز علـــى إمـــرار معـــاملات الحالات هذه المرة.



B- Solution algorithm

```
Compute_employee_pay
      Set valid_input to true
      Read emp_no, pay_rate, hrs_worked
      DOWHILE more records
         validate_input_field (valid_input)
```

استدعاء دالة التحقق (لاحظ المعامل)

```
         IF valid_input THEN
            Calculate_employee_pay
```

استدعاء دالة حساب الأجر

```
            Print emp_no, pay_rate, hrs_worked, employee_pay
         ELSE
```

```
        set valid_input to true
      ENDIF
      Read emp_no, pay_rate, hrs_worked
    ENDDO
END
```

validate_input_fields (valid_input) (لاحظ المعامل) دالة التحقق

```
    Set error_message to blank
    IF pay_rate > 25.00 THEN
      error_message = "Pay rate exceeds $25.00"
      valid_input = false
      Print emp_no, pay_rate, hrs_worked, error_message
    ENDIF
    IF hrs_worked > 60 THEN
      error_message = "Hours worked exceeds limit of 60"
      valid_input = false
      Print emp_no, pay_rate, hrs_worked, error_message
    ENDIF
END
Calculate_employee_pay
    IF hrs_worked <=35 THEN
      emp_pay = pay_rate*hrs_worked
    ELSE
      overtime_hrs = hrs_worked - 35
      overtime_pay = overtime_hrs*pay_rate*1.5
      emp_pay = pay_rate*35 + overtime_pay
      Print emp_no, pay_rate, hrs_worked, emp_pay
    ENDIF
END
```

4-8 التصميم الكائني

في ثنايا هذا الكتاب كررنا القول بأنه عليك أن تركز على ما يفعله البرنامج، أكثر من تركيزك على كيفية تنفيذه. فإذا ما تعقد البرنامج، فعليك محاولة تجزئته إلى عدد من الوظائف الأساسية،

وتجعل لكل وظيفة دالة مستقلة. وتسمى هذه المنهجية أسلوب التصميم من أعلى إلى أسفل. والتصميم الكائني يهدف أيضا إلى إجراء مثل هذه التجزئة، إلا أنه يجزئ البرنامج على صورة كائنات **objects** وليس دوال.

الكائنات

يمكن النظر للكائن على أنه إناء يحتوي على مجموعة من البيانات ومجموعة من العمليات تنفذ على تلك البيانات. وعلى ذلك، فإن الكائن يكون تعريفه كالتالي:

- اسم يعطى له.
- مجموعة من القيم، قد تكون على مهيكلة بحيث يكون للكائن أكثر من خصيصة.
- مجموعة من العمليات التي يمكن أن تنفذ على تلك القيم.
- وبعبارة أخرى، فالكائن هو أحد هياكل البيانات التي تعرف مصحوبة بالعمليات التي تقع عليها. فعلى سبيل المثال، قد ننظر لفهرس على أنه كائن، فنراه يتضمن:
- مجموعة من القيم، هي محتواته، والأرقام التي يبحث فيه بواسطتها.
- مجموعة من العمليات، وهي عمليات البحث والتنسيق والتعديل التي تجرى على محتوياته.

والمزية الأساسية للنظر إلى الفهرس على صورة كائن في برنامج ما، هي أن التعامل معه يكون على استقلال بالنسبة لأي هيكل يكون عليه بياناته، وعن أية دالة أخرى تتعامل معه.

وفي التصميم الكائني يمكن النظر للنظام ككل على أنه سلسلة من وحدات بنائية تتعامل مع الكائنات، دون الخوض في كيفية تشكيل هذه البيانات، أو تفاصيل العمليات التي تعمل على تلك البيانات. ومن الخصائص الجوهرية للتصميم الكائني أن كل كائن يتعامل مع غيره من الكائنات دون أن يهمه كيفية صياغته. فكل ما هو مطلوب منه معرفته بالنسبة لغيره من الكائنات هو وجودها، وأسلوب التعامل معها، أي نوعية البيانات المتبادلة بينها. وينطبق ذلك أيضا على كل من الدالة الأصلية والدوال الفرعية.

إخفاء البيانات

في التصميم الكائني، يكون هيكل البيانات للكائنات مخفيا عن غيرها، وهو ما يطلق عليه مبدأ "إخفاء البيانات data hiding"، وهو مبدأ جوهري في هذا الأسلوب. وقد وضع هذا المبدأ لأول مرة Parnas[i] عام 1972. هذا المبدأ ييسر استخدام الكائنات، حيث يحررك من ضرورة معرفة تكوينها الداخلي، أو كيفية إجراء العمليات عليها. وفي هذه الحالة يمكنك التركيز تماما على أسلوب التعامل بين الدوال والاتصال فيما بينها عن طريق المعاملات. ومن المصطلحات الأخرى التي تعبر عن إخفاء البيانات "الكبسلة encapsulation"، و"تجريد البيانات data abstraction".

ويمكن أن يضع التصميم الكائني بعض المحددات على عملية تصميم الوحدات البنائية:

- يجب أن تكون الوحدات مستقلة تماما لا تعتمد على أية عملية في أي قسم من البرنامج
- يمنع تماما المشاركة في البيانات عن طريق البيانات العامة، كوسيلة للاتصال بين أجزاء البرنامج

وليس من أغراض هذا الكتاب أن يغوص في موضوع التصميم الكائني، فهو أسلوب خاص بالبرامج المعقدة والضخمة، ولكن يجب عليك أن تكون على بينة من هذه المنهجية ومصطلحاتها.

ملخص الفصل

عرفنا في هذا الفصل أنواع البيانات والفرق بين الأنواع الأولية من البيانات وهياكل البيانات.
كما عرضنا لطرق الاتصال بين الوحدات، وعرفنا المتغيرات المحلية والعامة، ومفهوم مدى المتغير والمشاكل التي تنتج عن الإكثار من استخدام المتغيرات العامة.
وعرضنا لمفهوم المعاملات كوسيلة للاتصال بين الدوال وكيفية تمثيلها على خريطة هيكل البرنامج، وأعطينا مثالين لنوعي المعاملات.

وأخيرا عرضنا باختصار لمفهوم التصميم الكائني للبرامج كأسلوب حديث في فن البرمجة، وقدمنا شرحا لمصطلحه الرئيسي؛ الكبسلة أو إخفاء البيانات.

تدريبات

استخدم خطوات التصميم البنائي في وضع خوارزمات البرامج التالية:

1- صمم برنامجا يستحث ويتقبل الأجر السنوي ويحسب ضريبة الدخل للممولين، ويخرج قيمة هذه الضريبة طبقا للشرائح المبينة في السؤال.

Design a program that prompts and accepts an employee's annual incom, calculates and displays income tax according the following table:

Portion of Salary الشريحة	Income tax rate (%)
Less than 5.,000	Exempted
5,000 to < 10.000	6
10,000 to < 20.000	15
20,000 to < 30,000	20
30,000 to < 40,000	25
40,000 and above	30

2- صمم برنامجا يستحث ويتقبل أربعة أعداد ويرتبها ترتيبا تصاعديا، ويظهرها على الشاشة. يستخدم البرنامج دالة فرعية ترتب كل عددين معا كل مرة.

Design a program that prompts and accepts four numbers, sort them into ascending sequence and displays them on the screen. The program uses a module called Sort_two_numbers which is called by the mainline to sort two numbers at a time.

3- صمم برنامجا يستحث ويتقبل أربعة أرقام تعبر عن إحدى السنوات ويحسب إن كانت بسيطة أم كبيسة، ويظهر على الشاشة نتيجة الحساب، وكذا رسالة خطأ إذا لم تكن الأرقام أربعة أرقام.

Design a program which will prompt and accept a four-digit number of a year (e.g. 1990) and determine if it is a leap year and prints the

result. A message is to be produced if the number is not of four digits.

4- يراد تصميم برنامج يقرأ ملف المبيعات لشركة ويطبع مقدار العمولة المستحقة للبائعين. يحتوي كل سجل في الملف على بيانات رقم البائع واسمه ومقدار مبيعاته في الشهر طبقا لنظام الشرائح المعطى بالسؤال، ويخرج تقريرا يتضمن رقم البائع واسمه والعمولة المستحقة، مع العنوان الذي تقترحه للتقرير ولأعمدته.

Design a program that reads a sales file and calculates the commission owing to each salesperson. Each input record contains the number, name and volume of sales for the month. The commission rate varies according to sales volume as follows:

Sales volume ($)	Commission rate (%)
0 - 200	5
201- 1000	8
1001- 2000	10
above 2000	12

The commission is calculated as a combination of the above volumes, for example, the commission owing to a volume of $1200 is calculated as follows:

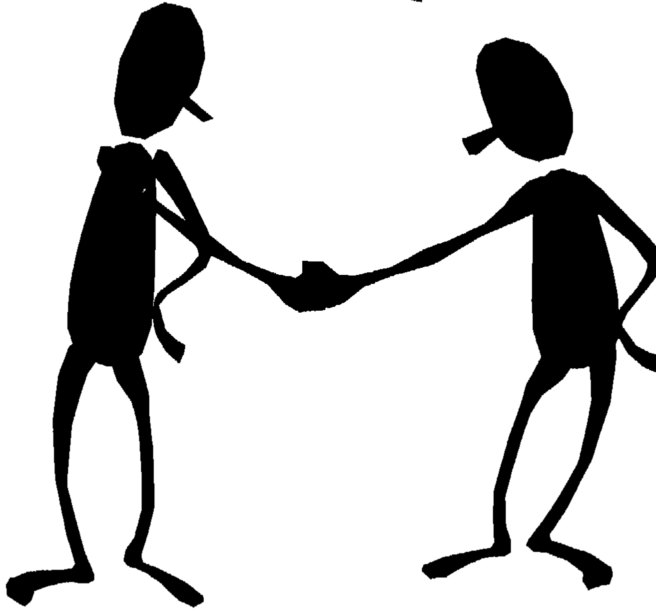
Commission = 200*0.05 + 800*0.08 + 1000*0.10 + (1200 - 2000)*0.12

The program is to print the salesperson's number, name, volume of sale and calculated commission whit appropriate headings.

[1] D. Parnas, "On the criteria to be used in decomposing systems into modules" Comm. ACM 15 (2), pp 1053-8 (1972)



الفصل التاسع الترابط والتقارن



الأهداف:

- ➢ التعريف بالترابط كمعيار لقوة بنيان البرنامج
- ➢ التعريف بالتقارن كمعيار لمدى إمكانية الاتصال بين أجزاء البرنامج

الخطوط العامة :

9-1 الترابط داخل الوحدات البنائية

9-2 التقارن بين الوحدات البنائية

9-3 ملخص الفصل

9-4 تدريبات

9-1 الترابط داخل الوحدات البنائية

لقد عرفنا الوحدة البنائية على أنها جزء من البرنامج مخصص لأداء وظيفة معينة. والوحدة البنائية تحتوي على مدخل واحد ومخرج واحد، ويجب اختيار اسم لها يعبر عن وظيفتها.

ويحتاج المبرمجون عادة لإرشادات تساعدهم على حسن تصميم الوحدات البنائية. فمن الأسئلة الشائعة بينهم "بأي طول تكون الوحدة البنائية؟"، "هل هذه الدالة غاية في القصر؟" أو "هل يجدر بي أن أضع كافة المدخلات في دالة واحدة؟"

وهناك وسيلة تعفيك من مثل هذه الهواجس، مبنية على مفهوم الترابط الداخلي للوحدة **cohesion**. هذا المفهوم يؤخذ على أنه معيار لمدى متانة بناء الوحدة، أو بعبارة أخرى إلى أي مدى تكون عناصرها أو تعابيرها مترابطة فيما بينها. فكلما زاد هذا الترابط، ازداد البنيان الداخلي للوحدة قوة.

وقد وضع Edward Yourdon and Larry Constatine[i] سبعة مستويات للترابط على شكل تدرج بحسب قوتها على النحو التالي:

Cohesion level	Cohesion attribute	Resultant module strength
Coincidental	Low Cohesion	Weakest
Logical	↓	↓
Temporal		
Procedural		
Communicational		
Sequential		
Functional	High Cohesion	Strongest

وسوف نعرض لكل مستوى من تلك المستويات.

الترابط العرضي

الترابط العرضي **Coincidental** هو أضعف صور الترابط، ويحدث حينما يكون تجميع العناصر في الوحدة البنائية بصورة عرضية، دون علاقة واضحة تجمع بينها. وفي هذه الحالة يصعب عمليا إيجاد وظيفة واضحة للوحدة. [الأصح القول بأنها حالة اللاترابط

ومن المؤكد أن هذا النوع من الترابط نادر هذه الأيام. وقد كان وجوده أصلا لأحد من هذه الأسباب:

- تقسيم برنامج ما بسبب متطلبات المكونات المادية، فيكون التقسيم اعتباطيا دون ضابط معين من ضوابط اعتبارات البرمجة.
- قد تقسم وحدات بطريقة تعسفية للتوافق مع قواعد موضوعة (مثلا، لا يزيد طول الوحدة عن 50 سطرا)
- قد تضم بعض الوحدات الصغيرة حتى لا تكون أقل من مقدار أدنى من الأسطر.

وقد اختفت هذه الاعتبارات تقريبا مع الزيادة المطردة في سعات التخزين وسرعة المعالجة.

وإليك مثال لمثل هذا النوع من الترابط:

```
File_processing
        Open employee updates file
        Read employee record
        Print page_heading
        Open employee master file
        Set page_count to one
        Set error_flag to false
END
```

ومن الواضح أنه لا توجد وظيفة منطقية تجمع بين كافة هذه الخطوات.

الترابط المنطقي

يحدث الترابط المنطقي **Logical** حينما تجمع عناصر الوحدة طبقا لنشاط معين، كأن تجمع كافة أوامر القراءة مثلا في وحدة واحدة، قد يطلق عليها Read_all_files. ولنضرب لذلك المثال التالي:

```
Read_all_files
        CASE of file_code
                1: Read customer transaction record
                        IF not EOF
                        increment customer_transaction_record
```

```
            ENDIF
        2: Read customer master record
            IF not EOF
            increment customer_master_record
            ENDIF
        3:  2: Read product master record
            IF not EOF
                increment prodect_master_record
            ENDIF
    ENDCASE
END
```

ومن هذا المثال تلاحظ أن الترابط هنا أقوى قليلا من الحالة السابقة، فعلى الأقل توجد رابطة ما تجمع بين الوحدات التي تنظمها العبارة CASE[كلها قراءة ملفات]، إلا أن ما ينفذ من الوحدة في كل حالة هو جزء صغير منها.

الترابط الزمني

يكون الترابط الزمني **temporal** حينما تكون الرابطة التي تجمع بين عناصر الوحدة البنائية هي الزمن، كأن تجمع كافة أنشطة الاستهلال أو الإنهاء في وحدة بنائية مستقلة، كما في المثال التالي:

```
Initialization
    Open transaction file
    Issue prompt "Enter today's date - DDMMYY"
    Read today_date
    Set transaction_count to zero
    Read transaction record
    IF not EOF
        increment transaction_count
    ENDIF
    Open report file
    Print_page_heading
    Set report_total to zero
END
```

هنا نجد الوحدة تحتوي على عناصر تمهد لعمل البرنامج [إدخال التـــاريخ، اســتهلال بعــض العدادات والمراكم بالصفر، طباعة رأس الصفحة...الخ].

وهنا تلاحظ أن الوحدة قد قامت بعدة أنشطة مختلفة.

الترابط الإجرائي

يحدث الترابط الإجرائي **procedural** حينما تجمع العناصر المختلفة في وحدة بنائية واحــدة لكون تلك العناصر تخضع لإجراء واحد، بمعنى أن أوامر الوحدة البنائية تنفذ بناء علــى تتــابع معين. ومن الأمثلة الواضحة على مثل هذا النوع من الوحدات البنائية هــو الدالــة الأصليــة للبرنامج، والتي تنفذ أوامرها طبقا لتتابع تنفيذ الدوال الفرعية.

ونقطة الضعف في الترابط الإجرائي هي أن الوحدة البنائية في حالته تخترق الحدود بين الوحدات البنائية الأخرى، بمعنى أنها تحتوي على جزءا من وظيفة ما، ثم عددا من وظائف أدنى. وإليــك مثالا يوضح ذلك:

```
Read_student_record_and_fees_received
        Set number_of_records to zero
        Set total_fees_received to zero
        Read student record
        DOWILE more records exist
                Add fees_received to total_fees_received
                Add 1 to number_of_records
                read student record
        ENDDO
END
```

[وفي مثالنا هذا نجد أن الوحدة تحتوي على جزء من أوامر التعامل مع ملفات الطلبة، ثم تجمــع المصاريف التي حصلت منهم]. ولعلك تلاحظ أن اسم الوحدة البنائية يحتوي على أداة العطــف "and"، وهو ما يوحي بأن الوحدة تؤدي أكثر من وظيفة ما من وظائف البرنامج.

الترابط الاتصالي

يكون الترابط الاتصالي **communicational** حينما تجمع أوامر الوحدة البنائية لكونها تعمل جميعها على بيان معين واحد. ويوجد هذا النوع من الترابط في البرامج التطبيقية الخاصة بمعالجة البيانات. ومن الأمثلة لذلك الوحدات التي تتعامل مع معالجة عمليات التحقق لكافة حقول السجل، أو معالجة أسطر التقرير تمهيدا لطباعته.

هذا الترابط أقوى من السابق لكون العلاقة التي تجمع عناصر الوحدة معتمدة على البيانات، وليس على الخطوات المتتابعة للتحكم في البرنامج. ولكن نقطة الضعف فيه تكمن في حقيقة أنه يتكون من خليط من عمليات المعالجة المتعلقة بأحد بيانات البرنامج، كما في المثال التالي:

```
Validate_prodect_record
      IF transaction_type NOT = "0" THEN
            error_flag = true
            error_message = "Invalid transaction type"
            print_error_report
      ENDIF
      IF customer_number is NOT numeric THEN
            error_flag = true
            error_message = "Invalid customer number"
            print_erro_report
      ENDIF
      IF product_no has a leading blank THEN
                        error_flag = true
            error_message = "Invalid transaction type"
            print_error_report
      ENDIF
            error_flag = true
            error_message = "Invalid product no"
            print_error_report
      ENDIF
END
```

وهكذا ترى أن كافة العناصر تعمـــل حـــول علميـــة التحقـــق، والـــتي تؤثـــر في البيـــانين error_flag وerror_message.

الترابط التتابعي

يحدث الترابط التتابعي **sequential** في حالة كون الوحدة البنائية تحتوي على عناصر يعتمـــد كل عنصر على نتيجة المعالجة في العنصر السابق عليه، بمعنى أن تكون مخرجات العنصر الســـابق هي مدخلات العنصر اللاحق. ويمكن تشبيه الوحدة البنائية هنا بخط تجميـــع: سلســـلة مـــن الخطوات التحويلية المتتابعة للبيانات. وإليك مثال لذلك:

```
Process_purchase
        Set total_purchases to zero
        Get number_of_purchases
        Do loop_index = 1 to number_of_purchases
                Get purchase
                add purchase to total_purchases
        ENDDO
        sales_tax = total_purchases*sales_tax_percent
        amount_due = total_purchases + sales_tax
END
```

ومن المثال تلاحظ أنه في هذه الوحدة قد تم حساب إجمالي المشتريات total_purchases ثم استخدمت النتيجة في حساب الضريبة sales_tax ثم استخدم البيانان في حســـاب إجمـــالي المستحق amount_due .

الترابط الوظيفي

يحدث الترابط الوظيفي عندما تتجه كافة عناصر الوحدة البنائية لتنفيذ مهمة معينة. وفي هـــذه الحالة يكون من السهل أن تعطى الوحدة اسما مكونا من فعل ومفعول به.

وتمثل الوحدات الموجهة لإجراء حسابات معينة مثالا طيبا للترابط الوظيفي، كما يوضح المثـــال التالي:

```
Calculate_sales_tax
```

```
        IF product is tax exempted THEN
                sales_tax = 0
        ELSE
                IF product_price < 50.00 THEN
                        sales_tax = product_price*0.25
                ELSE
                        IF product_price < 100.00 THEN
                                sales_tax = product_price*0.35
                        ELSE
                                sales_tax = product_price*0.50
                        ENDIF
                ENDIF
        ENDIF
END
```

ملخص مستويات الترابط

حينما تصمم برنامجا، عليك أن تحاول تجزئته إلى وحدات بنائية تهدف كل واحدة منها إلى تنفيذ مهمة معينة. وإذا ما تحقق مثل هذا الترابط الوظيفي فإن الوحدات تكون أكثر استقلالا، وأيسر فهما، وأسهل في عمليات التحديث المستقبلية.

وفي بعض الأحوال لا يكون متيسرا أن يتحقق لكافة الوحدات مثل هذا الترابط الوظيفي، فقد تكون لبعض الوحدات ترابط ذو مستوى أدنى، أو حتى خليط من بعض صور الترابط، وقد لا يمثل ذلك أي مشكلة، على أنه من الضروري أن تكون واعيا لمسألة الترابط، وأن يكون لجوؤك لمستوى أدنى مرتبة منه عن إدراك ولأسباب منطقية.

النقطة الجوهرية أن تتذكر دائما أن هدفك الرئيسي هو أن تصوغ برنامجا ووحداته البنائية بحيث تكون سهلة في التتبع والفهم، وأن يكون تحديثها المستقبلي ميسرا لمن يتولى ذلك من بعدك. وكلما كانت وحدات برنامجك أعلى ترابطا، تكون قد حققت هذا الهدف.

9-2 التقارن بين الوحدات البنائية

من المفروض أيضا عند التفكير في تصميم البرامج ألا يقتصر الأمر على اعتبار الترابط الداخلي للوحدات البنائية، بل أن يشمل أيضا سريان البيانات بين الوحدات. عليك أن تقلل ما أمكن من الاعتماد المتبادل بين الوحدات، محققا لها أكبر قدر من الاستقلالية. ويسمى الاتصال بين الوحدات "التقارن coupling"، أو "الواجهة interface".

ويعبر التقارن بين مدى تبادل البيانات بين الوحدات البنائية. فيعني التقارن القوي شدة تأثير هيكل وحدة ما على وحدة أخرى. وحيث إنه يوجد أكثر من مسار في البرنامج، فإن انتشار الأخطاء فيما بينها يكون له أكثر من طريق.

والتقارن الضعيف هو الصورة العكسية، فالوحدات ضعيفة التقارن فيما بينها تكون أكثر استقلالية وايسر في التعديل.

وقد وضع Glenford Myrs[ii] مقياسا للتقارن يشبه المقياس المقترح للترابط، حدد فيه خمسة مستويات من التقارن متدرجة من الأقوى، والذي يقابل أسوأ درجة في التصميم، إلى الأضعف، والذي يحقق أفضل نتائج تصميمية. والمقياس على الوجه التالي:

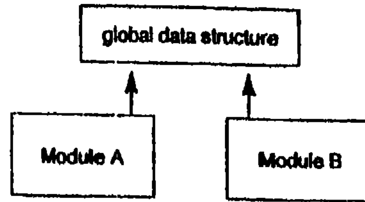
Coupling level	Coupling attribute	Resultant module design quality
Common	Tight coupling	poorest
External	↓	↓
Control		
Stamp		
Data	Loose coupling	best

ويجب الأخذ في الاعتبار أن هذا التصور ليس حاسما، فهو مجرد تصور لواضعه بالنسبة للموضوع.

التقارن الشائع

يكون التقارن شائعا **common** حينما تتشارك الوحدات جميعا في التعامل مع نفس هيكل البيانات (سجل أو مصفوفة)، والذي يكون في هذه الحالة من النوع العام global. معنى ذلك

أن البيان يمكن لكافة الدوال أن تتعامل معه وأن تغير فيه، وهو ما يجعل البرنامج صعب القـــراءة والتتبع.



ويبين البرنامج الهيكلي التالي دالتين بينهما تقارن مشترك، لكونهما قادرين على التغيير في نفـــس هيكل البيانات، وهو ملف العملاء.

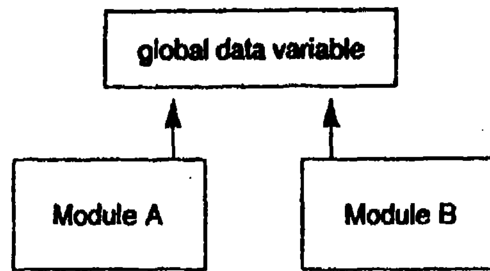
```
A Read_customer_record
        Read customer_record
                Read customer_record
                IF EOF THEN
                        set EOF_flag to true
                ENDIF
END
END
B Validate_customer_record
                IF customer_record is NOT mumeric THEN
                        error_message = "Invalid customer number
                ENDIF



        END
```

التقارن الخارجي

يحدث التقارن الخارجي **external** حينما يكون بوسع دالتين أو أكثر التعامل مع نفس المتغـــير العام، وهو يماثل التقارن الشائع غير أن التعامل يكون مع المتغيرات العامة وليس هيكلا للبيانات. وحيث إن المتغير هو هيكل أولي، فإن هذا التقارن يعتبر أقل شدة عن السابق.



والمثال التالي هو لدالتين بينهما تقارن خارجي، حيث يمكنهما أن يتعاملا مع نفس المتغير، sales_tax.

```
A Calculate_sales_tax
          IF product is sales exempted THEN
               sales_tax = 0
          ELSE
               IF product_price < 50.00
                    sales_tax = product_price*0.25



          ENDIF
     END

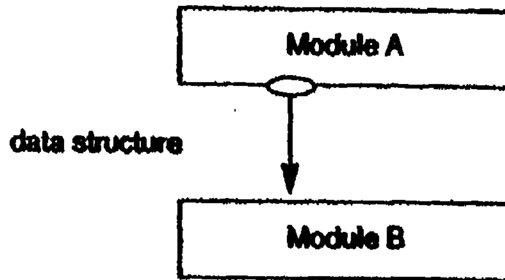
B Calculate_amount_due



          amount_due = total_amount + sales_tax
     END
```

تقارن التحكم

يكون تقارن التحكم control coupling حينما تقوم دالة بترحيل متغير التحكم إلى دالـــة أخرى ليقوم بتولي مهمة التحكم في تنفيذ الدالة الأخيرة. ويسمى متغير التحكم "راية flag" أو "سويتش switch" ويرحل بين الدوال كمعامل.



وإليك مثال لهذا التقارن:

```
A Process_input_code
          Read input_code
          Choose_appropreate_action



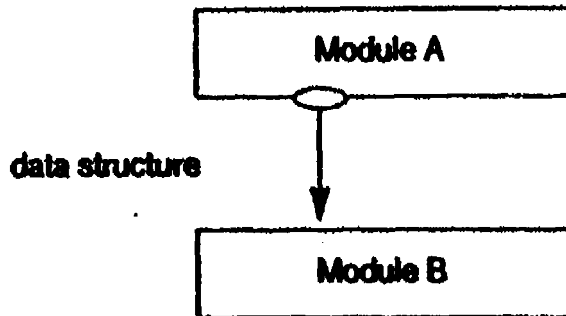

     END
B    Choose_appropreate_action
          CASE OF input_code
               1: Read employee record
               2: Print_page_heading
               3: Open employee master file
               4: Set page_count to zero
               5:   error_message  =  "Employee  number  not
numeric"
          ENDCASE
     END
```

وترى في هذا المثال أن الدالة الأولى قد قرأت كود الإدخال، ثم نقلت (رحلت) هـــذا الكـــود للدالة الثانية المسئولة عن تحديد التصرف بناء على الكود الذي تمت قراءته. [يقال أيضا أن الدالة الأولى قد استدعت الثانية للقيام بعملية التحكم].

ونقطة الضعف في هذا النوع من التحكم هي أن عملية إمرار المتغيرات الخاصة بالتحكم يعـــني ضرورة أن تكون دالة واعية بالتركيب الداخلي للدالة الأخرى.

التقارن بهياكل البيانات

في تقارن هيكل البيانات stamp coupling تمرر دالة هيكل بيانات غير عام، ويرحل الهيكل على هيئة معامل.



هذا النوع من التقارن غير قوي، ولذا فإنه يؤدي إلى درجة طيبة من تصميم الوحدات البنائية. فالعلاقة بين الوحدات تقتصر على الهيكل الذي يمرر، دون أن يكون ضروريا أن تعرف أية دالة شيئا عن التركيب الداخلي للدالة الأخرى. وإليك مثال لهذا النوع من التقارن، يمرر فيه هيكل البيانات current_record

```
A Proecdd_transaction_record



        IF transaction record is for a male THEN
              Process_male_student (current_record)
        ELSE
              Process_male_student (current_record)
        ENDIF



END


B Process_male_student (current_record)
        Increment male_student_count
```

```
IF student_age >21 THEN
        increment mature_male_count
ENDIF




END
```

التقارن بالبيانات

يحدث هذا التقارن data coupling حين ترحل دالة متغيرا غير عام إلى دالة أخرى، وهو يمثل أضعف أنواع التقارن، وبالتالي أكثر مرونة للوحدات وأفضل تصميم لها. وإليك مثال لهذا النوع من التقارن.

```
A Process_customer_record


        Calculate_sales_tax (total_price, sales_tax)


END


B Calculate_sales_tax (total_price, sales_tax)
        IF total_price < 10.00 THEN
                sales_tax = total_price*0.25
        ELSE
                IF total_price < 100.00 THEN
                        sales_tax = total_price*0.3
                ELSE
                        sales_tax = total_price*0.4
                ENDIF
        ENDIF
```

END

ملخص الفصل

قدمنا في هذا الفصل عددا من الموضوعات المختلفة التي يجب أخذها في الاعتبار عند تصميم البرامج. فالبرنامج جيد التصميم هو البرنامج الذي تتمتع وحداته البنائية بأكبر قدر متاح من الاستقلالية، وأن تكون سهلة القراءة والتتبع والتحديث. وتتميز مثل هذه الوحدات بقوة الترابط الداخلي وضعف التقارن المتبادل.

والتقارن هو معيار متانة البنيان الداخلي للوحدات البنائية، وقد قدمنا سبعة مستويات للتقارن متدرجة من الأضعف إلى الأقوى، وأعطينا أمثلة على كل مستوى.

كما أن التقارن معيار لمدى التأثير المتبادل للوحدات على بعضها البعض، وأيضا قدمنا خمسة مستويات متدرجة من الأقوى للأضعف، أي متدرجة بترتيب تصاعدي لجودة التصميم. وقدمنا لأمثلة توضيحية لهذه المستويات.

تدريبات

بفضل ما حصلت عليه من معلومات في هذا الفصل، يطلب منك مراجعة الخوارزميات التي صممت في الفصلين السابقين لدراستها من حيث قوة الترابط وضعف التقارن، وتطويرها ليكون الاتصال بينها عن طريق إمرار المعاملات التي تعطي أكبر قدر من الاستقلالية للوحدات.

---

[i] Edward Yourdon and Larry Constatine, *Structured Design: Fundamentals of a Decipline of Computer Program and System Design*, Prentice-Hall, 1979.

[ii] Glenford Myre, *Composite Structure Design*, Van Nostrand Reinhold, 1978.



الفصل العاشر تطبيقات متقدمة



الأهداف:

- تقديم خوارزم البرامج الهيكلية لأهم خمسة تطبيقات في الحياة العملية:
- إنتاج التقارير متعددة الصفحات
- انتاج التقارير بالمجاميع التحليلية
- انتاج التقارير بالمجاميع التحليلية متعددة المستوى
- تحديث الملفات التتابعية
- معالجة المصفوفات

الخطوط العامة :

10-1 الهيكل الأساسي للبرامج

10-2 إنتاج التقارير متعددة الصفحات

10-2 انتاج التقارير بالمجاميع التحليلية

10-3 انتاج التقارير بالمجاميع التحليلية متعددة المستوى

10-4 تحديث الملفات التتابعية

10-5 معالجة المصفوفات

10-1 الهيكل الأساسي للبرامج

الهدف من هذا الفصل هو عرض الصورة العامة لخوارزمات بمجموعة مختارة من التطبيقات العملية. وسوف تتضمن هذه التطبيقات كافة الخصائص التي عرضنا لها في الفصول التسعة السابقة. ويحقق كل حل من الحلول المعروضة تنظيما هيكليا عالي الترابط.

ولتبسيط العرض، جعلنا الوحدات تتعامل مع بعضها البعض من خلال بيانات عامة. على أنه يمكن بسهولة تعديلها ليكون التعامل من خلال إمرار المعاملات على الصورة التي شرحناها في الفصل الثامن.

وقد كان الخط العام في الكتاب مبنيا على الاعتماد على حل عام لخوارزم معالجة الملفات التتابعية sequential files، ويمكن اعتبار هذا الخوارزم بمثابة العمود الفقري، يأخذ برنامجه الهيكلي الصورة التالية:

```
Process_sequential_file
 Initial processing
 Read first record
 DOWILE not EOF
  Process this record
  Read next record
 ENDDO
 Final processing
END
```

وتمثل هذا الصورة العامة للخوارزم إطارا عاما لكل التطبيقات التجارية تقريبا. وهو لا يحتوي معالجة الصفحات المتعددة، ولا إخراج المجاميع. ويمكن توسيع هذا الحل العام ليحتوي على متطلبات أي موضوع برمجي معين.

وسوف نستخدمه بدورنا ولضع الخوارزمات الخاصة بالأمثلة الخمسة التي نقدمها.

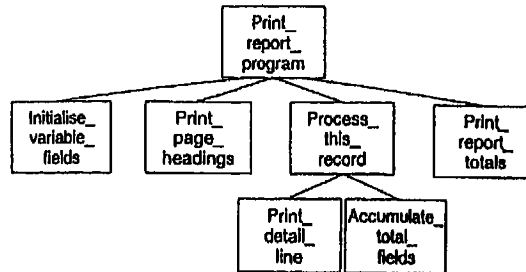
10-2 إنتاج التقارير متعددة الصفحات

تحتوي أغلب التقارير على عنوان للتقرير، وعنوان للحقول، وأسطر البيانات، وأسطر للمجاميع. كما يطلب من البرنامج أن ينتقل من صفحة إلى أخرى بعد عدد معين من الأسطر. وقد يكـون جزء من تقرير نمطي على الصورة التالية:

12/05/96	CLOTHING CO CURRENT ACCOUNT REPORT GLAD RAGS.		PAGE: 1
CUATOMER NUMBER	CUSTOMER NAME	CUSTOMER ADDRESS	ACCOUNT BALANCE
12345	Sportly's Buotique	The Mall, Redfern	300.00
12346	Slinkey's Nightrear	145 Picnic Roud	400.50
		Total customers on file	200
		Total customers with balance owing	150
		Total balance owint	4300.00

يمكننا تطوير الصورة العامة لمعالجة الملفات التتابعية لتحقق هذه المتطلبات الخاصة بإنتاج التقـارير على الوجه التالي:

A Hierarchy chart



وترى أن وحدة Process-this-record قد طورت لتكون إحدى وحدات الوحدة البنائيـة الأشمل، وهي Print_report_program. وينعكس هذا التطور على الخوارزم العام كالآتي:

B Solution algorithm

Mainline الدالة الأصلية

Print_report_program

Initialize_variable_fields استدعاء دالة الاستهلال

Read first record

DOWILE not EOF

IF line coount > max_detail_lines تحديد عدد أسطر الصفحة

Print_page_heading استدعاء دالة طباعة العنوان

set line_count to zero وضع عداد أسطر الصفحة في الصفر

ENDIF

Process this record

Read next record

ENDDO

Subordinate modules الدوال الفرعية /

(المستوى الأول)

1 Initialize_variable_fields دالة استهلال القيم الابتدائية

set accumulators to zero

set page_count to zero

set line_count to zero

set max_detail_lines to designed value

END

2 Print_page_heading

increment page_count

print heading lines

print column heading lines

print blank line (if required)

END

3 Process_this_record

Perform necissary calculations (if any)

Print_detail_line

Accumulate_total_fields

END

4- Print_detail_fields

build detaild line

print detail line

increment line_count

END

المستوى الثاني

```
5- Accumulate_total_fields
 increment accumulators as required
 END
6Print_report_totals
 build total line(s)
 print total line(s)
 END
```

ويمكن استخدام هذا البرنامج الهيكلي كإطار لخوارزم إنتاج التقارير التي تتضمن تغييرا في الصفحات.

3-10 إنتاج التقارير بالمجاميع التحليلية

إنتاج التقارير التي تحتوي على مجاميع تحليلية شائعة في البرامج التطبيقية التجارية. فسطر المجموع التحليلي هو السطر الذي يظهر بمجموع خاص بفئة معينة في الملف مميزة بحقل مفتاحي key field. ويحسب المجموع كلما حدث تغيير في قيمة حقل مفتاحي.

وفي مثل هذه التقارير، تتوقف عملية المعالجة كلما تغير الحقل المفتاحي، لحساب المجموع المطلوب، [ولذا فيطلق على هذا النوع من البرامج "برامج التوقف المحكوم control break" ويسمى المجموع التحليلي: المجموع المحكوم control total].

والبرامج التي تخرج بمجاميع تحليلية تكون إما على مستوى واحد single-level، أو متعددة المستوى multi-level. ويعتمد ذلك على كون السجل مفهرس بناء على حقل مفتاحي واحد أو أكثر من حقل مفتاحي.

وإليك جدول ذو مجموع تحليلي وحيد المستوى:

	PC-COMPUTER CO.				
12/05/96	SALES REPORT BY SALESPERSONS			PAGE: 1	
SALESPER ON	SALESPERO N NAME	PRODUCT NUMBER	QTY SOLD	PRICE	AMO UNT

	PC-COMPUTER CO.				
NUMBER					
1001	Mary Smith	1032	2	10.00	20.00
		1033	2	20.00	40.00
		1044	2	30.00	60.00
			Sales total for Mary Smith		120.00
1002	Jane Brown	1032	2	10.00	20.00
		1045	1	35.00	35.00
			Sales toal for Jane Brown		55.00
			Report sales total		175.00

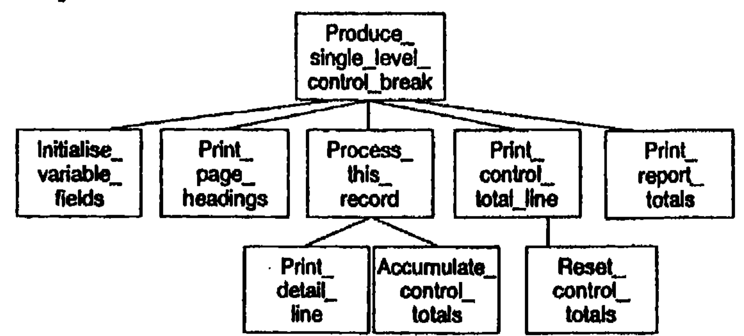
وتلاحظ أن المجموع حسب عندما تغير الحقل المفتاحي، وهو رقم البائع، من 1001 إلى 1002.

وهناك أمران يجب الانتباه إليهما عند تصميم برنامج ينفذ إخراج المجاميع التحليلية:

- يجب أن يكون الملف قد تمت ترتيبه بناء على حقول مفتاحية، (في المثال المعطى قد رتب الملف تصاعديا طبقا لرقم البائع).
- في كل مرة يقرأ سجل ما من ملف الإدخال، تقارن قيمة الحقل المفتاحي مع القيمة السابقة، بحيث يحسب المجموع كلما تغيرت تلك القيمة، وقبل أن تجرى معالجة السجل الحالي.

ويمكن تطوير خوارزم إنتاج التقرير ليشمل وحدتين جديدتين لتنفيذ المجاميع التحليلية، هما Print_control_total-line المسئولة عن طباعة المجموع التحليلي، وReset_control_totals الخاصة بإفراغ مركم المجموع التحليلي. هذه الدالة الأخيرة تكون دالة فرعية من الدالة الأولى، فيكون الشكل التخطيطي للبرنامج على الوجه التالي:

A- Hieracchy chart



يتطلب برنامج إخراج المجاميع التحليلية ما يلي:

- متغير باسم this_control_field يحفظ فيه اسم الحقل المفتاحي الذي قرأ لتوه
- متغير باسم prev_control_field يحفظ فيه اسم الحقل المفتاحي السابق. (بعد قـــراءة أول سجل، تجعل قيمة كلا المتغيرين هي قيمة الحقل المفتاحي لهذا السجل)
- الحقول المطلوبة لتخزين المجاميع التحليلية
- متغير لتخزين المجموع الكلي

B- Solution algorithm

```
Mainline
Process_single_level_control_break
 Initialize variable_fields
 Print_page_heading
 Read first record
 this_control_field = control field
 prev_control_field = control field
 DOWHILD not EOF
  IF this_control_field = prev_control_field THEN
   print_control_total_line استدعاء دالة طباعة المجموع التحليلي
   prev_control_field = this_control_field
  ENDIF
  IF line coount > max_detail_lines
   Print_page_heading
   set line_count to zero
  ENDIF
```

```
 Process this record
 Read next record
 this_control_field = control field
ENDDO
Print_control_total-line استدعاء دالة طباعة المجموع التحليلي
Print_report_totals
END
```

ويجب مراعاة أربعة نقاط حتى يسير برنامج المجاميع التحليلية سيرا صحيحا:

كلما قرأ سجل جديد، يأخذ المتغير this_control_field قيمة الحقل المفتاحي للسجل الجديد.

عند قراءة أول سجل، يعطى المتغيرات this_control_field وprev_control_field قيمة الحقل المفتاحي له، ويمنع ذلك أن تجري عملية طباعة المجموع قبل معالجة السجل الأول.

تحدّث قيمة المتغير وprev_control_field كلما أحس البرنامج بتغير في قيمة الحقل المفتاحي

عند الوصول لأخر الملف، تستدعى الدالة الخاصة بطباعة المجموع التحليلي مرة أخرى، إما لطباعة أخر مجموع تحليلي، أو لإرجاع حالة السجلات set the records.

10-انتاج التقارير بالمجاميع التحليلية متعددة المستوى

قد يكون الملف المعالج مفهرسا على أكثر من حقل مفتاحي، ففي المثال التالي ينتج تقرير المبيعات بتجميع لمبيعات البائعين، ثم تجمع بجاميع الإدرات التابعة لها البائعون، فيكون التجميع بناء على حقل رقم البائع وحقل رقم الإدراة معا. وإليك تقرير من هذا القبيل:

		PC-COMPUTERS CO.				
12/12/96		SALES REPORT BYSALESPERSONS AND DEPARTEMENTS				PAGE: `
DEPT	NUMBER	NAME	PRODUCT	QTY	PRICE	AMOUNT

		PC-COMPUTERS CO.				
01	1001	Mary Smith	1032	2	10.00	20.00
			1033`	2	20.00	40.00
			1044	2	30.00	60.00
			Sales total for Mary Smith			120.00
	1002	Jane Sown	1032	2	10.00	20.00
			1045	1	35.00	35.00
			Sales total for Jane Sown			55.00
			Sales total for Dept. 01			175.00
02	1050	Jem Pond	1033	2	20.00	40.00
			1044	2	30.00	60.00
			Sales total for Jem Pond			100.00
			Sales total for Dept. 02			100.00
			Report Sales total			275.00

ومن التقرير المبين تلاحظ أن المجموع يحسب كلما تغير رقم الإدارة وكلما تغير رقم البائع، إذ يوجد لدينا حقلين مفتاحيين رتب على أساسهما الملف. وتنطبق المفاهيم المذكورة في حالة برامج المجاميع التحليلية ذات المستوى الواحد على البرامج متعددة المستويات:

فيجب أن تكون مدخلات الملف مرتبة طبقا للحقول المفتاحية، وتنظم هذه الحقول بحيث يكون هناك حقل أشمل major control field يضم الحقول الأدنى minor control fields.

ففي مثالنا تعتبر الإدارات حقلا أشمل يضم حقول البائعين التابعين لكل إدراة.

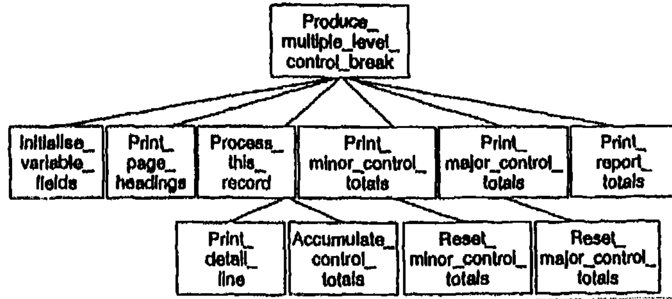
وفي كل مرة يقرأ فيها سجل من الملف، تقارن الحقول المفتاحية، فإذا ما تغير حقل مفتاحي أدنى، طبع المجموع الخاص بالحقل السابق، وإذا ما تغير حقل اشمل، تطبع مجاميع الحقل الشامل السابق والحقول الأدنى السابقة أيضا.

ويطور خوارزم المستوى الواحد ليضم الوحدتين الإضافيتين للبرنامج ذي المستويين المبين، فإذا ما كان البرنامج ذو ثلاث مستويات، تضاف وحدتان أخريتان، وهكذا.

وقد تغيرت أسماء الوحدات التي تنتج المجاميع لكي تميز بين مجاميع الحقول الشاملة والحقول الأدنى، فأعطيت تلك الوحدات الأسماء: Print_minor_control_totals لطباعة الحقول

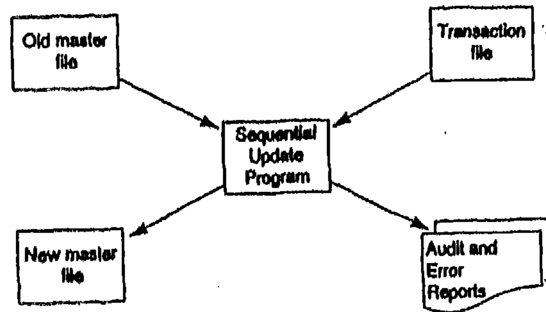
الأدنى، وPrint_major_control_totals لطباعة الحقول الشاملة. وينطبق نفس الشـــيء علـــــى دوال الإرجـــــــاع، فـــــــأصبحت Reset_minor_controls, .Reset_major_control_totals

A- Hierarchy chart



5-10 تحديث الملفات التتابعية

تحديث الملفات التتابعية من أهم عمليات المعالجة بالدفعات batch processing. ويتضمـــن ذلك تحديث الملف الأساسي master file بواسطة ملف المعـــاملات transactional file. وتكون كلا الملفين من النوع التتابعي، وينتج من هذه العملية ملف رئيسي جديد يحتوى علـــى آخر بيانات ملف المعاملات. ومن المعتاد أن يخرج في هذه العملية تقرير مراجعة وتقرير أخطـــاء. ويكون الشكل العام لهذه العملية كالتالي:



مفاهيم النظام

1- الملف الرئيسي

الملف الرئيسي master file هو الملف الذي يحتوي على البيانات الدائمة أو قليلة التعديل، وتكون سجلات هذه الملفات مرتبة تتابعيا طبقا لحقل معين، أو عدة حقول معينة. ومن أمثلة هذا النوع من الملفات ملف العملاء الذي يحتوي على أسم العميل وعنوانه ورقم الهاتف له وحد الائتمان الممنوح له ورصيده.

2- ملف المعاملات

يحتوي **ملف المعاملات transaction file** على بيانات النشاط الجاري والتي تؤدي إلى تحديث الملف الرئيسي. ويكون ذلك بإجراء ما يلي:

- إضافة سجل جديد
- تعديل سجل موجود
- محو سجل موجود

فعلى سبيل المثال، قد يتضمن ملف معاملات العملاء بيانات مقصود بها محو سجل عميل أو إضافة سجل جديد أو تعديل في سجل قائم.

وتكون ملفات المعاملات مرتبة تتابعيا بنفس الحقول المفتاحية التي للملفات الرئيسية.

3- تقرير المراجعة

يحتوي تقرير المراجعة على العمليات التي أجريت على الملف الرئيسي، وهو ما يسهل عملية التتبع المحاسبي audit trail فيما بعد.

4- تقرير الأخطاء

ويحتوي على الأخطاء التي قد تقابل أثناء عملية تحديث الملف الرئيسي، كأن يراد حذف سجل غير موجود، أو إضافة سجل موجود بالفعل.

منطق التحديث التتابعي

يعتبر المنطق الذي يسير عليه برنامج تحديث لملفات تتابعية أعقد منه للبرامج الأخرى، حيث يتطلب ملفين في عملية الإدخال، وتسير عملية المعالجة على أساس قراءة سجل من كل ملف، ومقارنة الحقلين المفتاحيين، ونتيجة لهذه المقارنة تكون المعالجة على ثلاثة صور:

إذا كانت قيمة الحقل المفتاحي لملف المعاملات أقل من قيمتها في الملف الرئيسي، فإن المعاملة هي في الغالب إضافة لسجل جديد، وتكون بيانات سجل ملف التعاملات هي التي سوف تنقل للملف الرئيسي الجديد. ويقرأ السجل التالي في ملف المعاملات.

إذا كانت قيمة الحقل المفتاحي لسجل المعاملات والملف الرئيسي متساويتان، فإن العملية تكون إما تعديلا أو محوا، وفي الحالة الأولى يتم تعديل بيانات الملف الرئيسي ليعكس التعديل الجديد في البيانات. أما إذا كانت العملية محوا، لا يكتب السجل في الملف الرئيسي الجديد. ويقرأ السجل التالي في ملف المعاملات.

إذا كان الحقل المفتاحي أكبر، يكتب السجل القديم للملف الرئيسي في الملف الجديد بلا تغيير، ويقرأ السجل التالي في الملف الرئيسي لإجراء المقارنة مع سجل ملف المعاملات.

ويجب أن تحتوي برامج معالجة الملفات التتابعية على جزء يتيح تعامل سجل الملف الرئيسي مع أكثر من سجل معاملات. كما يجب أن يتضمن البرنامج مواجهة حالة وجود سجلات خاطئة وتحدث الأخطاء في الحالات التالية:

محاولة إضافة سجل ذي حقل مفتاحي له مقابل في الملف الرئيسي.

محاولة تعديل سجل لا يوجد له مقابل في الملف الرئيسي.

محاولة محو سجل لا يوجد له مقابل في الملف الرئيسي.

محاولة محو سجل لم يصل الرصيد فيه للصفر.

خوارزم الخط المتوازن

شغل خوارزم التحديث التتابعي عقول المفكرين في علم البرمجة لسنوات، وقدم المؤلفون العديد من الحلول، لم يكن أي منها حلا عاما بالمعنى الحقيقي، بل كانت في الغالب حلولا خاصة بلغات برمجية معينة.

وقد قدم Barry Dwyer خوارزما يمثل حلا عاما جيدا للمسألة[i]. وقد عرف هذا الحل بـ "**The Balance Line Algorithm**"، أو "**خوارزم الخط المتوازن**". ويمكن لهذا الخوارزم أن يتعامل مع مسألة عدة ملفات معاملات تحدث ملفا رئيسيا، وكذا حالة وجود الأخطاء.

ونقدم فيما يلي البناء الهيكلي لهذا الخوارزم، والذي يعتمد على مفهوم "السجل المختار" أو السجل الجاري "current record"، وهو السجل الذي يجري اختياره بناء على مقارنة الحقل المفتاحي لكلا الملفين، الرئيسي والمعاملات، ليكون جاهزا إما للتعديل او لينقل إلى الملف الرئيسي الجديد. ويعرف السجل تحت المعالجة بأنه *السجل الذي يكون له أقل قيمة للحقل المفتاحي من بين الملفين الرئيسي وملف المعاملات*. وتكون هيئة ذلك السجل record format هي هيئة سجل الملف الرئيسي الجديد.

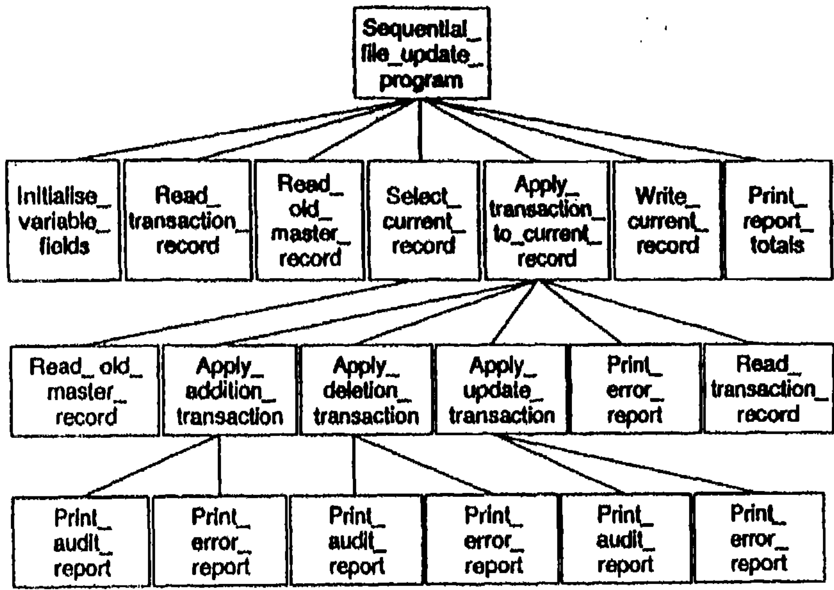
وعلى ذلك، فإذا كان رقم سجل ملف المعاملات هو الأقل، كان سجل ملف المعاملات هو السجل المختار للمعالجة، أما إذا كانت القيمة مساوية أو أكبر، كان سجل الملف الرئيسي هو المختار للمعالجة.

ويظل السجل المختار للمعالجة هو الخاضع لعمليات المعالجة إلى أن تنتهي تلك العمليات، فيجري نقله للملف الرئيسي الجديد، ثم يجري اخيار سجل جديد.

ويحتاج الخوارزم إلى متغير باسم current_record_status يستخدم كراية flag تحدد كون السجل المختار جاهزا للمعالجة أم لا. فإذا ما كان هذا المتغير فعالا active دل ذلك على ان السجل قد تم اختياره وأصبح جاهزا للتعديل أو للكتابة في الملف الجديد، أما إذا كان المتغير غير فعال inactive، دل ذلك على أن السجل غير متاح، ربما لأنه تم التأشير عليه للمحو.

ويستمر التعامل مع الملفين حتى نصل إلى حالـــة "نهايـــة العمـــل"، والـــتي تعـــرف بالمتغـــير end_of_job وذلك حين لا يصبح لدى كلا الملفين بيانات إدخال متاحة. وحيث أننـــا لا نعرف مسبقا أي من الملفين سوف يصل لنهاية الملف EOF أولا، فإن الملـــف فـــإن الحقـــل المفتاحي في أي من الملفين يعطى قيمة عالية عند الوصول لنهاية الملف. وحين يصل حقـــل في ملف لهذه القيمة العالية تستمر معالجة الملف الآخر طبقا لما هو مقدر حتى يصل حقله المفتـــاحي لنفس القيمة، فتكون حالة تحقق end_of_job هي وصول الملفين معا إلى هذه القيمة المرتفعة.

لنضع الآن الخوارزم العام لبرنامج معالجة ملف رئيسي تتابعي، على ضوء الشكل البنائي المبين



[يحتاج الخوارزم العام للدوال المبينة في الشكل، وإلى المتغيرات التالية:

Variable	Use
current_record_status	flag to determine availability of current record
total_ transaction_records	counter for transaction records
total_old_master_rec ords	counter for old master records
total_new_master_rec ords	counter for new master records

Variable	Use
total_error_records	counter for error records
end_of_job	flag for program termination
transaction_record_key	stores value of key field of transaction record
old_master_record_key	stores value of key field of old master record
error_message	stores error message
transaction_type	stores transaction type

ولتتذكر أن المتغيرات التي تعمل عمل راية flag تكون من النوع المنطقي]

أولا: الحالة الأصلية

```
Mainline
 Sequential_update_program
 Initialize_variable_fields
 Read_transaction_record
 Read_old_master_record
 set current_record_status to "inactive"
 DOWHILE Not end_of_job
  Select_current_record
  DOWHILE transaction_record_key = current_record_key
   Apply_transaction_to_current_record
  ENDDO
  IF current_record_status = "active" THEN
   Write_current_record
   set current_record_status to "inactive"
  ENDIF
 ENDDO
 print_report_totals
END
```

ثانيا: الدوال الفرعية

Subordinate modules

```
1- Initialize_variable_fields
 set_total_transaction_records to zero
 set_total_old_master_records to zero
 set_total_new_master_records to zero
 set_total_error_records to zero
 set end_of_job to false
END
2- Read_transaction_record
  Read transaction record
  IF not EOF THEN
   increment total_transaction_records
  ELSE
   set transaction_record_key to high value
   IF old_master_record_key = high value
    set end_of_job to true
   ENDIF
  ENDIF
 END
3- Read_old_master_record
  Read old master record
  IF not EOF THEN
   increment total_ old _ master _records
  ELSE
   set old _ master _records _key to high value
   IF transaction_record_key = high value
    set end_of_job to true
   ENDIF
  ENDIF
 END
4- Select_current_record
  IF transaction_record_key < old _ master _records _key THEN
   set up current record with transaction record fields
```

```
  ELSE
   set up current record with old master record fields
   set current_record_status to active
   Read_old_master_record
  ENDIF
 END
5- Apply_transaction_to_current_record
  CASE OF transaction_type
   addition: Apply_addition_transaction
   deletion: Apply_ deletion _transaction
   update: Apply_ update _transaction
   other: error_message = "invalid transaction type"
    Print_error_message
  ENDCASE
 END
6- Write_current_record
  write current record to new master file
  increment total_new_master_records
 END
7- print_report_totals
  print total_transacton_records
  print total_old_master_records
  print total_new_master_records
  print total_error_records
END
8-  Apply_addition_transaction
 IF current_record_status = "inactive THEN
 set current_record_status to active
 Print_audit_report
 ELSE
  error_message = "Invalid additio; record already exists"
  Print_error_report
 ENDIF
END
```

```
9- Apply_deletion_transacton
 IF current_record_status = "active THEN
 set current_record_status to inactive
 Print_audit_report
 ELSE
  error_message = "Invalid deletion; record not on master file"
  Print_error_report
 ENDIF
END
10- Apply_update_transaction
 IF current_record_status = "active THEN
 apply required change(s) to current record
 Print_audit_report
 ELSE
  error_message = "Invalid update; record not on master file"
  Print_error_report
 ENDIF
END
11- Print_audit_report
 print transzctin details on audit report
 CASE OF transactin_type
  addition: print "record added"
  deletion: print "record deleted"
  u[date: print "record updeted"
 ENDCASE
END
12- Print_error_report
 print transzctin details on error report
 print error_message
 increment total_error_records
END
```

وتلاحظ أن البرنامج ينتهي حينما تصل قيمتي الحقلين المفتاحين في الملف الرئيسي وملف المعاملات إلى القيمة العليا في نفس الوقت.

4-10 معالجة المصفوفات

المصفوفة هي هيكل للبيانات مكونة من عدد من المتغيرات التي تكون من نفس النوع، مثلا، درجات امتحان طلاب.

وكل بيان مختزن في المصفوفة يعرف بأنه "**عنصر element**" من عناصر المصفوفة، ولكافة العناصر نفس الاسم، وهو اسم المصفوفة ذاتها، وتتميز العناصر عن بعضها برقم يسمى "**دليل المصفوفة array index**"، يوضع بعد الاسم أو تحته، ويشير لموضع (ترتيب) العنصر في المصفوفة.

وتلعب المصفوفات دورا هاما للغاية في تيسير التعامل مع مجموعات البيانات، ومن ثم فإنه يجدر بك أن تكون ملما بما يجري عليها من عمليات. وسوف نقدم لك في هذا القسم أهم تلك العمليات، وهي:

- تخزين قيم في المصفوفة.
- قراءة قيم عناصر من المصفوفة
- البحث في المصفوفة
  - ◊ البحث الخطي (المباشر)
  - ◊ البحث الثنائي (البحث بالتنصيف)

وفي الأمثلة التالية سوف تسمى المصفوفة "array"، ويطلق على الدليل index، والقيمة المدخلة input value. ويخزن العدد الأقصى الممكن تخزينه في المصفوفة في المتغير max_num.

تخزين قيم في المصفوفة

يطلق على تخزين القيم في المصفوفة "القــراءة في المصفوفــة reading values into the array". ويمكن القيام بذلك عن طريق دوارة بسيطة، تنتهي باكتمال المصفوفة أو بانتهاء القيــم المدخلة.

```
Read_values_into_array
 set max_num to designated value تحديد عدد عناصر المصفوفة
 set index to zero
 read first input value قراءة أول قيمة
 DOWHILE (input value exists AND index < max_num)
  index = index +1 تحديد العنصر الأول
  array(index) = input value تخزين القيمة الأولي في العنصر الأول
  read next input value
 ENDDO
END
```

قراءة عناصر المصفوفة

يطلق على هذه العملية "الكتابة إلى خارج المصفوفة writing out of an array". وتتم هذه العملية عن طريق دوارة بسيطة.

```
Write_values_of_array
 set index to 1
 DOWHILE index <= max_num
  write array(index)
  index = index +1
 ENDDO
 END
```

البحث الخطي في المصفوفة

من العمليات الشائعة في التعامل مع المصفوفات البحث فيها عن بيان معين. ويعـــني "البحـــث الخطي (أو البحث المباشر linear search) أن ينظر في كل عنصر من عناصر المصفوفة واحدا

بعد الآخر، بدءا من أول عنصر فيها، لاختبار توافقه مع القيمة المنشودة. وينتهي البحث إمـــا بوجود العنصر المتوافق، أو بانتهاء عناصر المصفوفة.

ويحتاج خوارزم البحث الخطي متغيرا نسميه **element_found** يعمل كراية تبين الوصـــول للعنصر المنشود من عدمه. وتختزن القيمة المنشودة في متغير نسميه **input_value**. كما نختـزن عدد عناصر المصفوفة في المتغير **max_num** كالمعتاد.

```
Linear_search_of_array
 set max_num to designated value
 set element_found to false
 set index to 1 البدء بأول عنصر
 DOWHILE (NOT element_found AND index < max_num)
  IF array(index) = input_value THEN
   set element_found to true
  ELSE
   index = index +1
  ENDIF
 ENDDO
END
```

البحث الثنائي في المصفوفات

حينما يكون عدد المصفوفات كبيرا، والعناصر مرتبة تصاعديا، يمكن الإسراع من عملية البحث بالطريقة التالية، والتي تسمى "**البحث الثنائي binary search**":

يحدد منتصف المصفوفة، ويختبر إن كان هو العنصر المتوافق، أم لا.

إذا كان العنصر المنصف ليس هو المطلوب، فينظر إذا كان أقل أم أكبر.

في حالة كون العنصر المنصف أكبر، ينصف الجزء السفلي، ويختبر إن كان هو المتوافق.

في حالة كون العنصر المنصف أقل، ينصف الجزء العلوي من المصفوفة، ويجرى نفس الاختبار.

تتكرر عملية التنصيف والاختبار إلى أن نصل للعنصر المتوافق، أو تنتهي عناصر المصفوفة.

وفي الخوارزم التالي نصنع راية باسم element_found والمتغير max_num، ثم متغـــيرين، low_element للإشـــارة للموضـــع الســـفلي للنصـــف المختـــبر (الأدنى في الـــترتيب) وhigh_element للإشارة للقمته (الأعلى في الترتيب).

ويستمر البحث إلى أن نجد العنصر المنشود أو تتوقف عمليـــة التنصيـــف، أي تكـــون قيمـــة low_element ليست أقل من قيمة high_element، (أي أكبر منها أو تساويها)، ويـــدل ذلك على انتهاء عناصر المصفوفة (وصلت عملية التنصيف إلى العنصر الأول أو الأخير).

```
Binary_search_of_array
 set element_found to false
 set low_element to 1
 set high_element to max_num
 DOWHILE (NOT element_found AND low_element <= high_element)
  index = (low_element+high_element)/2 تحديد العنصر المنصف للمصفوفة
  IF input_value = array(index) THEN
   set element_found to true
  ELSE
   IF input_value < array(index)
    high_element = index-1
   ELSE
    low_element = index +1
   ENDIF
  ENDIF
 ENDDO
END
```

خص الفصل

صص هذا الفصل لوضع خمسة من أشهر الخوارزمات التي يقابلها المبرمج في الحياة العمليـــة، ويمكن استخدامها كنمط للعمليات التي تعالجها في البرامج المختلفة بـــالتعديل الـــذي يتطلبـــه البرنامج الواقعي.

تدريبات

1- صمم برنامجا يقرأ الملف الرئيسي لشركة ملابس ويخرج التقرير المبين، والخاص بالعملاء الذين يوجد لهم رصيد في الشركة.

Design a program that reads Customer Master File of a clothing company, and prints a report of all customer records which have an account balance greater than zero.
The master file contains customer's number, name, address (street, city, state and postcode) and account balance.
The output report, which contains 3 totals and 35 lines per page is as follows:

	GLAD RAGE CLOTHEIG CO. CUURENT ACCOUNT BALANCES		PAGE
CUSTOMER NO.	CUSTOMER NAME	ADDRESS	ACCOUNT BALANCE
XXXX	XXXX	XXXXX	9,999.99
XXXX	XXXX	XXXXX	9,999.99
		total customers on file	999
		Total customers with balance owing	999
		Total balance owing	9,999.99

2- يراد تصميم برنامج يقرأ ملف المبيعات لشركة مرتب تتباعيا طبقا لرقم البائع، ويمكن أن يكون لكل بائع أكثر من سجل بحسب البضائع التي يتعامل معها، ثم يخرج تقريرا بالمبيعات يتضمن إيراد البائعين (الكمية المباعة*ثمن الوحدة) ويطبع الإجماليات حسب ما هو مبين في السؤال.

Design a program that reads a Sales File and produces a Sales report for a company. The Sales report shows the details of each sale made per salesperson, and total of sales for that person.

The fields of the Sales File is: salesperson's number, name, product number, quantity sold for the month and unit price. There may be more than one record for the salespersons according to the products they have sold. The Sales File is sorted sequentially according to salesperson's number.
Your program should read the file sequentially, calculate the total sales per product, total sale per salesperson and total sales of report.
The report layout is:

	MULTI-DISK COMPUTER CO.				
DATE: XX/XX/XX	SALES MONTHLY REPORT				PAGE:
SALESPERSON NO.	NAME	PRODUCT NO.	QTY SOLD	UNIT PRICE	TOTAL
XXXX	XXXXXXXXXX	XXXX	99	999.99	9,999.99
		XXXX	99	999.99	9,999.99
			total for xxxx		9,999.99
			Report total		99,999.99

3- في المسألة السابقة يراد تطوير البرنامج لكي يحتوي على توزيع البـائعين علـى الإدارات، حيث أضيف حقل رقم الإدارة لملف المبيعات، ويحسب إجمالي إيراد الإدارات.

The same Sales File as described in Problem 2, with addition of a further field; department number. The Sales report has been modified accordingly as follows:

		MULTI-DISK COMPUTER CO.				
	DATE: XX/XX/XX	SALES MONTHLY REPORT				PAGE:
DEP NO.	SALESPERSON NO.	NAME	PRODUCT NO.	QTY SOLD	UNIT PRICE	TOTAL
XX	XXXX	XXXXXXXXXX	XXXX	99	999.99	9,999.99
			XXXX	99	999.99	9,999.99
				total for XXXX		9,999.99
				total for Dep XX		
				Report total		99,999.99

4- يراد وضع برنامج لتحديث الملف الرئيسي للعملاء لشركة، من واقـع ملـف معـاملات العملاء. يحتوي ملف المعاملات على نفس حقول الملف الرئيسي، مضافا إليه حقل بـه كـود عملية التحديث المطلوبة لكل سجل، على ما هو مبين بالسؤال. وينتج البرنامج الملف الرئيسـي الجديد، بالإضافة إلى تقرير المراجعة وتقرير الأخطاء بالرسائل المبينة.

A company requires a program to update its Customer Master File according to a sequential file of update transactions as input file.
The master file contains the customer number, name, address, and account balance. The transaction file contains the same fields, as well as a transaction code of 'A' for add, 'D' for delete and 'u' for update.
Both files are sorted into customer number. There can be multiple updates for one record. A new customer file is to be produced.
Transaction records are to be processed as follows:
1 If transaction record is an Add, the transaction is to be written to the new master file.
2 If transaction record is a Delete, the old customer record with the same number is not written on the new file.
3 If transaction record is an Update, the old record with the same number is to be updated as follows:
If customer name is present, update customer name.

◊ If street is present, update street.
◊ If town is present, update town.
◊ If city is present, update city.
◊ If postcode is present, update postcode.
◊ If balance is present, subtract balance paid from amount balance on old file.

As each transaction is processed, the transaction details are to be printed on the Customer Audit Report, with message: 'record added', 'record deleted' or 'record updated' as applicable.
If a transaction record is in error, the transaction details are to be printed on the Customer Error Report, with one of the following messages:
'Invalid addition, customer already exists'.
'Invalid deletion, customer not on file'
'Invalid update, customer not on file'

5- يراد من برنامج التحقق من كود المنتج في ملف معاملات المنتجات. المطلوب تخزين أكـواد عد تخزين الأكواد في المصفوفة، يقرأ الملف تتابعيا، ويتم التحقق من كود المنتج بالبحث عن هذا

الكود في المصفوفة. إذا كان البحث ناجحا، لا يطلب أي إجراء، وإذا كان الكود غير موجـود، يكتب السجل في تقرير للأخطاء، مع رسالة تبين الخطأ. مطلوب أن يتضمـن التقريـر أيضـا مجموع الأكواد الخاطئة.

A program is required to validate the product code on a Product Transaction File. A file of valid codes is to be read at first into an array of 300 elements at the beginning of the program.
Once the codes have been set up in the array, the transaction file is read sequentially. The product code on each transaction record is to be validated by searching the array for that code. If code is found, no action is required. If code is not found, the record is to be written to an Error Report, with message: 'Invalid code'. Total of error codes is to written at the end of the report.

[1] Barry Dwyer, "One More Time - How to Update a Master File". Comm. ACM, Vol. 124, No. 1 January 1981.

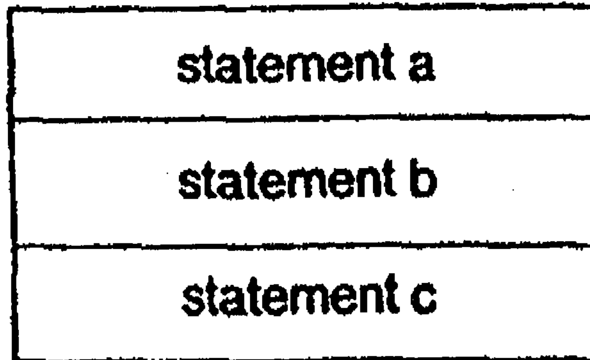
ملحق 1- أشكال ناسي-شنايدر

تقدم أشكال ناسي-شنايدر أسلوبا تصويريا لمن يفضلون هذا النوع من تمثيل خوارزمات البرامج، وسوف نعرض في هذا الملحق للخوارزمات التي تمت صياغتها بأسلوب اللغة شبه البرمجية ممثلة بهذا الأسلوب.

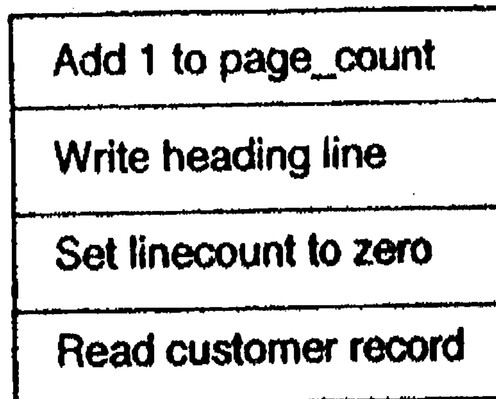
الهياكل الحاكمة:

1- الأوامر المتتابعة

تمثل الأوامر المتتابعة في أشكال ناسي-شنايدر على الصورة التالية:

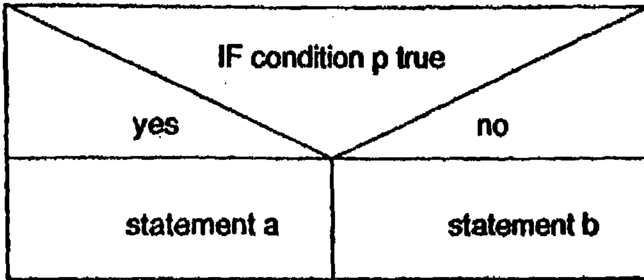


وذلك كما في المثال التالي:

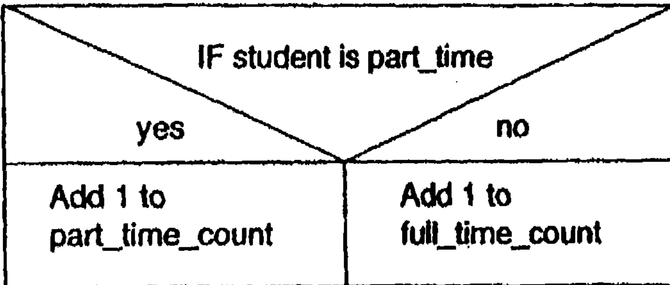


2- القرارات

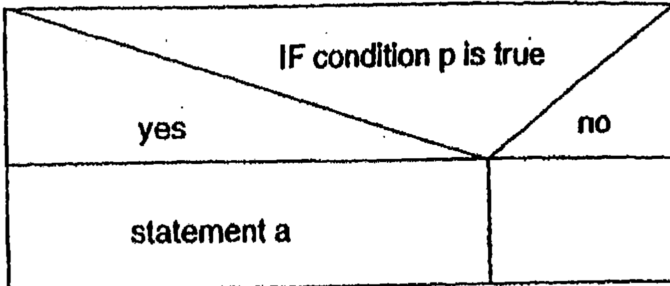
تمثل القرارات في أشكال ناسي-شنايدر على الصورة التالية:



كما في المثال التالي:

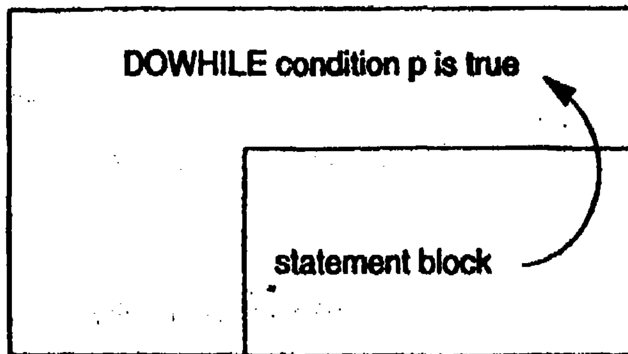


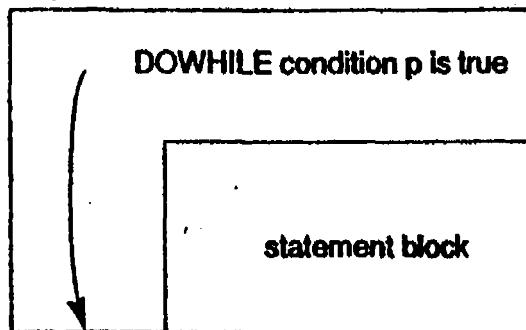
أما العبارة غير المتفرعة فتمثل بالشكل التالي:



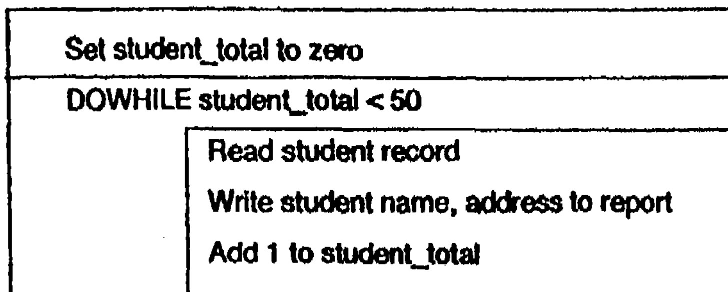
3- التكرار:

يمثل هيكل التكرار في أشكال ناسي-شنايدر على الصورة التالية:





وفي هذا الشكل تنفذ الأوامر في الصندوق الداخلي، والذي يمثل آخر أمر فيه المحدد delimiter للدوارة، حيث بعده يعاد اختبار شرط التكرار، فيظل قائما طالما كان الشرط متحققا (السهم إلى أعلى). وحينما يتغير وضع الشرط من التحقق إلى عدم التحقق، يخرج البرنامج من الدوارة (السهم إلى أسفل). والمثال التالي يبين ذلك:



أمثلة

أ- الأوامر المتتابعة

يمكن تصوير الأمثلة المعطاة في فصل 3 بأشكال نانسي-شنايدر، وسوف نأخذ منـــها المثــال التالي:

مثال 3-3 حسابه وقته قص الحشائش

مطلوب برنامج يحسب المدة الزمنية اللازمة لقص حشائش حديقة منزل، باستخدام بيانات عــن أبعاد المساحة الكلية، وأبعاد مساحة المباني، ووقت قص المتر المربع. وإليك صياغة البرنامج مـــع التحليل:

A program is required to read the length and width of the block, and the length and width of the house which is built on the block. The algorithm should compute and display the time required to cut the grass around the house, at a rate of 2 square meters per minute.

A- Defining deagram

Input	Processing	Output
block_length	Prompt for measurements	mowing_time
block_width	Get block measurements	
house_length	Calculate house measurements	
house_width	Calculate mowing time	
	Display mowing time	

B- Solution algorithm

Calculate_mowing_time
Prompt user for block_length, block_width
Get block_length, block_width
Get block_length, block_width
block_area = block_length* block_width

Prompt user for house_length, house_width
Get house_length, house_width
house_area = house_length* house_width
mowing_area = block_area - house_area
mowing_time = mowing_area/2
Output mowing_time to screen

ب- القرارات

يمكن تصوير الأمثلة المعطاة في فصل 4 بأشكال نانسي-شنايدر، وتكون على الصورة التالية:

عبارة IF البسيطة

IF account_balance < $300.00	
yes	no
service_charge = $5.00	service_charge = $2.00

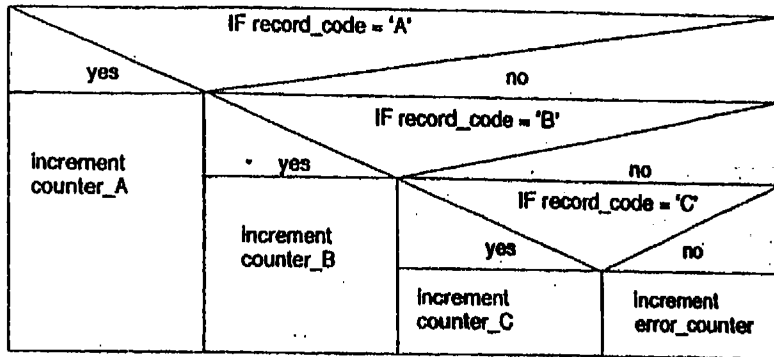
عبارة IF غير المتفرعة

IF student_attendance = part_time	
yes	no
add 1 to part_time_count	

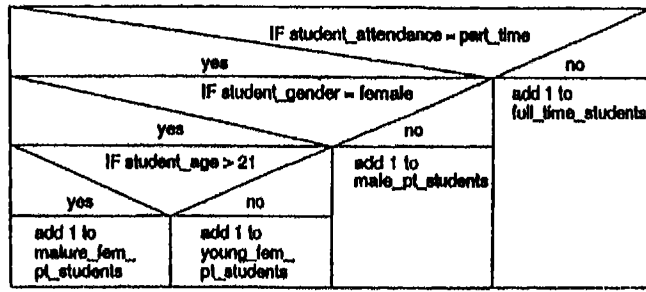
عبارة IF المركبة

IF student_attendance = part_time AND student_gender = female	
yes	no
add 1 to fem_part_time_count	

عبارة IF المتداخلة



عبارة IF المتداخلة غير خطيا



مثال 4-1 ترتيب ثلاثة أحرف

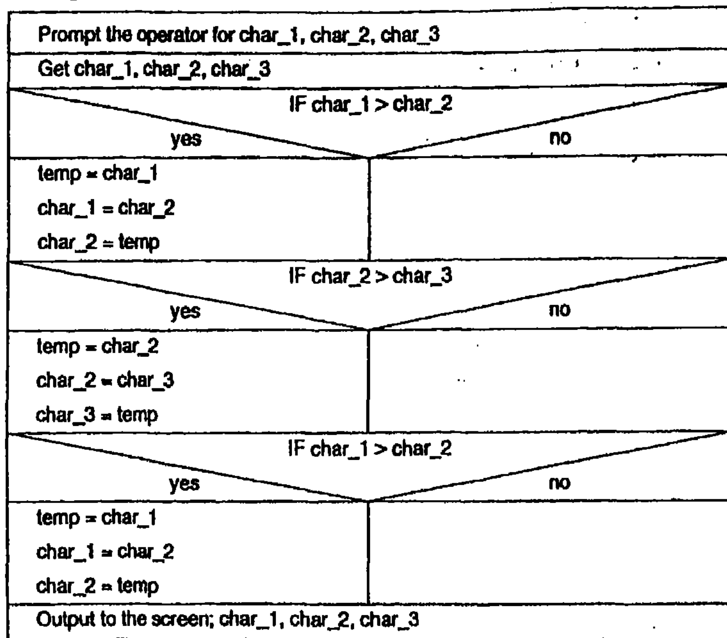
المطلوب تصميم برنامج يتقبل ثلاثة أحرف ويرتبهم تصاعديا.

Design a program which will prompt the user for three characters, accept those characters as inputs, sort them into ascending sequence, and output them on the screen.

A- defining diagram

Input	Processing	Output
char_1	Prompt for characters	char_1
char_2	Accept three characters	char_2
char_3	Sort three characters	char_3
	Output three characters	

B- Solution algorithm



مثال 4-2 معالجة سجل عميل

يراد وضع برنامج يتلقى اسم العميل، ومقدار طلبيته، والكود الضريبي، ويحسب القيمة الإجمالية للفاتورة.

A program is required to <u>read</u> a customer's name, a purchase amount and a tax code which will be one of the following:

0 tax exempted
1 tax 3%
2 tax 5%
3 tax 7%

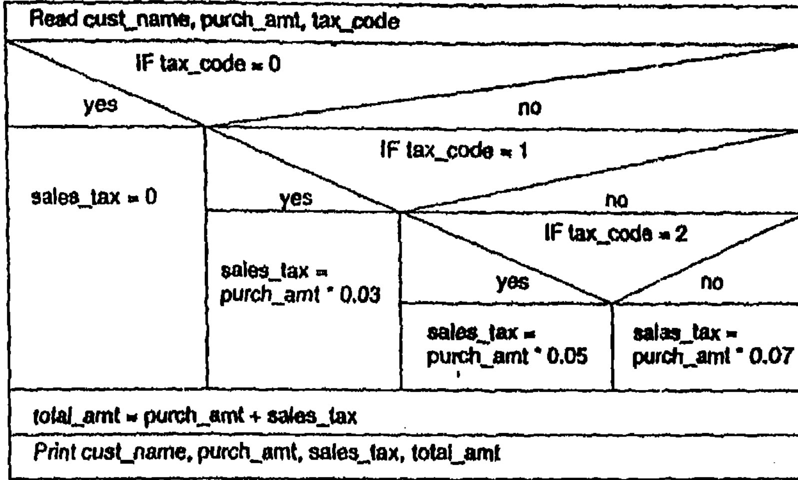
The program must then <u>compute</u> the sales tax and total amount due and <u>print</u> the customer's name, purchase amount, sales tax and total amount due.

A- Defining diagram

Input	Processing	Output
cust_name	Read customer details	cust_name
purch_amt	Compute sales tax	purch_amt

Input	Processing	Output
tax_code	Compute total amount	sales_tax
	Print customer details	total_amt

B- Solution algorithm



مثال 4-3 حساب الأجر

تريد شركتك أن تضع لها برنامجا يتقبل رقم الموظف والأجر في الساعة وعدد ساعات العمل في الأسبوع، ويحسب أجر العامل، علما بأن الحد الأقصى لساعات عمل العامل هي 60 ساعة أسبوعيا، والحد الأقصى لأجر الساعة 25 جنيها، وعدد الساعات المعتادة هي 35 ساعة أسبوعيا. ويحسب الأجر الإضافي بواقع مرة ونصف الأجر المعتاد.

A program is required to read an employee's number, pay rate, and the number of hours worked in a week. The program is then to compute the employee's weekly pay and print it along with the input data.

According to the company's rules, no employee may work more than 60 hours per week, and the maximum hourly rate is $25.00 per hour. If more than 35 hours are worked, then payment for the overtime hours worked is calculated at time-and-a-half. If the hours worked field or the hourly rate is out of range, then the input data and an error message are printed without calculation made.

A- Defining diagram

Input	Processing	Output
emp_no	Read employee details	emp_no
pay_rate	Validate input fields .	pay_rate
hrs_worked	Calculate employee pay	hrs_worked
	Print employee details	emp_pay
		error_mssage

B- Solution algorithm

Set valid_input_fields to true

Set error_message to blank

Read emp_no, pay_rate, hrs_worked

IF pay_rate > $25.00

yes:
error_message = 'Pay rate exceeds $25.00'
valid_input_fields = false
Print emp_no, pay_rate, hrs_worked, error_message

no

IF hrs_worked > 60

yes:
error_message = 'Hours worked exceeds limit of 60'
valid_input_fields = false
Print emp_no, pay_rate, hrs_worked, error_message

no

IF valid_input_fields

yes:

IF hrs_worked ≤ 35

yes:
emp_weekly_pay = pay_rate * hrs_worked

no:
overtime_hrs = hrs_worked – 35
overtime_pay = overtime_hrs * pay_rate * 1.5
emp_weekly_pay = (pay_rate * 35) + overtime_pay

Print emp_no, pay_rate, hrs_worked, emp_weekly_pay

no

عبارة CASE

تصور العبارة CASE في أشكال نانسي-شنايدر على الصورة التالية

CASE OF single variable

value 1	value 2	value n	value other
statement block_1	statement block_2	statement block_n	statement block_other

مثال 4-4 معالجة سجل عميل

A program is required to read a customer's name, a purchase amount and a tax code which will be one of the following:

0 tax exempted

1 tax 3%

2 tax 5%

3 tax 7%

The program must then compute the sales tax and total amount due and print the customer's name, purchase amount, sales tax and total amount due.

A- Defining diagram

Input	Processing	Output
cust_name	Read customer details	cust_name
purch_amt	Compute sales tax	purch_amt
tax_code	Compute total amount	sales_tax
	Print customer details	total_amt

B- Solution algorithm

```
Read cust_name, purch_amt, tax_code
CASE OF tax_code
  0: sales_tax = 0
  1: sales_tax = purch_amt * 0.03
  2: sales_tax = purch_amt * 0.05
  3: sales_tax = purch_amt * 0.07
total_amt = purch_amt + sales_tax
Print cust_name, purch_amt, sales_tax, total_amt
```

ج- التكرار

مثال 1-5 تحويل درجة الحرارة

لدينا محطة رصد لدرجات الحرارة تتلقى يوميا خمسة عشر درجة بالمقياس الفهرنهيتي، ويراد وضع برنامج لتحويلها إلى درجات حرارة مئوية، وإظهارها على شاشة الجهاز، ثم إظهار رسالة تفيد الانتهاء من عملية الرصد.

Every day, a weather station receives 15 temperatures expressed in degrees Fahrenheit. A program is to be written which will accept each Fahrenheit temperatures, convert it to Celsius and display the converted temperatures to the screen. After 15 temperatures has been processed, the message "All temperatures has been processed" is to be displayed on the screen.

Input	Processing	Output
f_temp	Get Fahreheit temperatures	c_temp
(15 temperatures	Convert temperatures	
	Display Celius temperatures	
	Display message	

B- Solution algorithm

```
Set temperature_count to zero
DOWHILE temperature_count < 15
        Prompt operator for f_temp
        Get f_temp
        Compute c_temp = (f_temp - 32) * 5/9
        Display c_temp
        Add 1 to temperature_count
Display 'All temperatures processed' to the screen
```

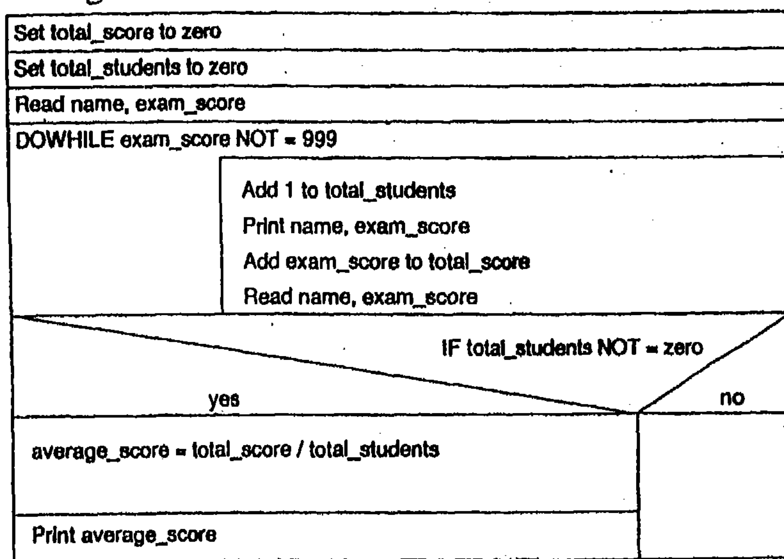
مثال 5-2 طباعة نتائج اختبار-معالجة البيانات التتابعية

A program is required to read and print a series of names and exam scores for students, the class average is to be computed and printed at the end of the report. Scores can range from 0 to 100. The last record contains a blank name and a score of 999 and should not be included in the calculation.

A- Defining diagram

Input	Processing	Output
name	Read student detail	name
exam_score	Print student detail	exam_score
	Compute average score	avg_score
	Pront average score	

B- Solution algorithm



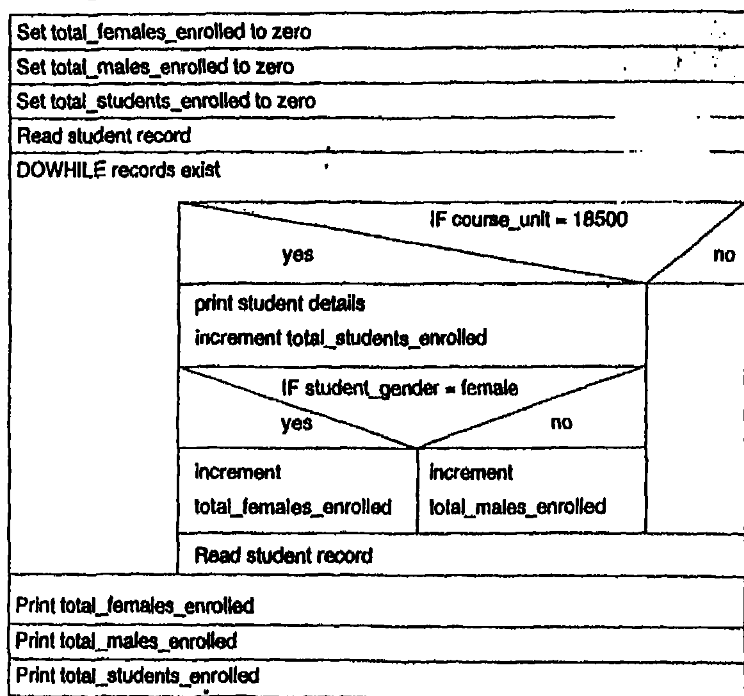
مثال 5-3 معالجة تسجيل الطلاب

A program is required which will read a file of students records, and select and print only those students enrolled in a unit named Programming 1. Each student record contains student number, name, address, postcode, gender and course unit number. The course unit number for Programming 1 is 18500. Three totals are to be printed at the end of report; total females, total males and total students.

A- Defining diagram

Input	Processing	Output
student_record	Read student records	selected students
. student_no	Select student records	records
. name	Print selected records	totals
. address	Compute total females	
. postcode	Compute total males	
. gender	Compute total students	
. course_unit	Print totals	

B- Solution algoritm



مثال 4-5 معالجة أصناف مخزنية

A program is required to read a series of inventory records that contain items number, item description and stock figure. The last record in the file has an item number 0. The program is to produce a "Low Stock Items" report, by printing only those records which have stock figure less than 20 items. A heading is to be printed at the top of the report and a total low stock item count to be printed at the end.

A- Defining diagram

Input	Processing	Output
Inventory record	Read inventory records	heading
. item_number	Select low stock items	selected records
. item_descrip	Print low stock records	. item_number
. stock_figure	Print total low stock records	. item_descrip
		. stock_figure
		total_low_stock

Solution algorithm

Set total_low_stock_items to zero
Print 'Low Stock Items' heading
Read inventory record
DOWHILE item_number > zero
- IF stock_figure < 20
  - yes:
    - Print item_number, item_description, stock_figure
    - Increment total_low_stock_items
  - no:
- Read inventory record

Print total_low_stock_items

ملحق 2- خوارزميات خاصة

الخطوط العامة

خوارزمات ترتيب البيانات

- الترتيب الفقاعي
- الترتيب بأسلوب الإدخال
- الترتيب بأسلوب الاختيار
- خوارزم التعامل مع المصفوفات
- هياكل البيانات الديناميكية
- الصفوف
- المكادس
- القوائم المترابطة

يحتوي هذا الملحق على خوارزمات لم نتعرض لها في صلب الكتاب ولكن قد تحتاج لها في المستقبل في حياتك العملية.

وأول مجموعة تحتوي على ثلاثة خوارزمات لترتيب البيانات: **الترتيب الفقاعي bubble sort** والترتيب بأسلوب الإدخال **insertion sort** والترتيب بأسلوب الاختيار **selection sort**. وتحتوي المجموعة الثانية على خمسة خوارزمات تتعامل مع المصفوفات. أما المجموعة الأخيرة فتقدم لك ثلاثة هياكل ديناميكية للبيانات، **الصفوف queues** والمكادس **stacks** والقوائم المترابطة **linked lists**، كما نقدم الخوارزمات التي تتعامل معها.

خوارزمات ترتيب البيانات

تقوم الخوارزمات في الأمثلة التالية بترتيب مصفوفة من أعداد صحيحة ترتيبا تصاعديا. وسوف تستخدم فيها التسميات التالية:

array: اسم المصفوفة

number_of_elements:عدد عناصر المصفوفة

temp: متغير لتخزين الأعداد مؤقتا خلال عملية التبديل

i: دليل الدوارة الخارجية

j: دليل الدوارة الداخلية

نفترض أنه قد تم تخزين المصفوفة array بعدد number_of_elements من العناصر:

الترتيب الفقاعي

يقوم الخوارزم في أسلوب الترتيب الفقاعي بمقارنة عددين متجاورين في المصفوفة في كل دورة. إذا لم يكن العددان مرتبين تصاعديا، يعكس ترتيبهما، وإلا يتركان على وضعهما. وعلى ذلك فإن الدورة الأولى تتسبب في جعل أكبر الأرقام في أسفل المصفوفة.

في الدورة الثانية يتعامل الخوارزم مع الأعداد الباقية، وعند انتهائها يكون أكبر الأعداد الباقية فوق العدد الأكبر مباشرة. وهكذا. [يمكن تصور الأعداد بعد كل دورة وقد طفت الأعداد الأصغر لأعلى، كما تفعل فقاعات الغازات داخل السوائل، وهو السبب في تسمية هذا الأسلوب].

وسوف نحتاج في هذا الخوارزم للمتغير المنطقي التالي:

elements_ switched: راية تبين حدوث عملية ترتيب في الدورة الجارية من عدمه.

```
Bable_sort_algorithm
        set i to number_of_elements
        set elements_switched to true
        DOWHILE (elements_switched AND i>=2)
                set j to 1
                set elements_switched to false
                DOWHILE j<= i-1
                        IF array(j) > array(j+1) THEN
                                temp = array(j)
                                array(j) = array(j+1)
                                array(j+1) = temp
```

```
                element_switced = true
            ENDIF
            j = j+1
        ENDDO
        I = I+1
    ENDDO
END
```

الترتيب بأسلوب الإدخال

تقوم فكرة الخوارزم باستخدام أسلوب الإدخال **insertion method** على مسح عناصر المصفوفة واحدا بعد الآخر، وفي كل مرة يحدث انعكاس للترتيب تتوقف عملية المسح مؤقتا لإجراء عملية مسح عكسية لوضع العنصر في مكانه من الترتيب، ثم تستأنف عملية البحث. وتزاح العناصر التي يتجاوزها المسح العكسي لكي تفسح مكانا للعنصر الذي سوف يدخل فيما بينها. وهذه الطريقة أكثر سرعة وكفاءة من الطريقة السابقة للترتيب.

```
Insertion_sort_algorithm
    set i to 1
    DOWHILE i <= (number_of_elements -1)
        IF array(i) > array(i+1) THEN
            temp = array(i+1)
            j=1
            DOWHILE (j>= 1 AND array(j) > temp
                array (j+1) = array(j)
                j = j-1
            ENDDO
            array(j + 1) = temp
        ENDIF
        i = i+1
    ENDDO
END
```

الترتيب بأسلوب الاختيار

تقوم فكرة هذا الخوارزم على الحصول على أصغر عدد في كل دورة، ثم تبديل وضعه مع العدد الذي في أعلى المصفوفة. بهذه الطريقة سوف ترتب المصفوفة بعد عدد من الدورات يساوي عدد عناصر المصفوفة -1.

وسوف نحتاج لهذا الخوارزم إلى المتغيرين التاليين:

smallest_element: أصغر أعداد المصفوفة يكتشف كل دورة

current_smallest_position: المكان الذي يجب أن يوضع فيه أصغر أعداد المصفوفة في الدورة الحالية.

```
Selection_sort_algorithm
        set current_smallest_position to 1
        DOWHILE                current_smallest_position               <=
(number_of_elements -1)
                set i to current_smallest_position
                smallest_element = array(i)
                set j = i+1
                DOWHILE j <= number_of_elements
                        IF array(j) < THEN
                                i = j
                                smallest_element = array(i)
                        ENDIF
                        j = j+1
                ENDDO
                array(i) = array(current_smallest_position)
                array(current_smallest_position) = smallest_element
        ENDDO
END
```

خوارزمات التعامل مع المصفوفات

إيجاد مجموع عناصر المصفوفة

```
Calculate_sum_of_array
        set sum to zero متغير كمركم لتخزين المجموع
        set i to 1
        DOWHILE i <= number_of_elements
                sum = sum + array(i)
                i = i + 1
        ENDDO
        print sum
END
```

إيجاد أصغر عنصر

```
Find_largest_element
        set i = 1
        set largest_element to array(1) وضع قيمة أول عنصر في المتغير المبين
        DOWHILE i < number_of_elements
                IF array(i+1) > largest_element THEN
                        largest_element=array(i+1)
                ENDIF
                i = i+1
        ENDDO
        print largest_element
END
```

إيجاد مدى المصفوفة

```
Find_range_of_elemenst
        set i = 1
        set largest_element to array(1)
        set smallest_element to array(1)
        DOWHILE i < number_of_elements
                IF array(i+1) > smallest_element THEN
```

```
                smallest_element=array(i+1)
            ELSE
                IF array(i+1) > largest_element THEN
                    largest_element=array(i+1)
                ENDIF
            ENDIF
            i = i+1
        ENDDO
        print ramge as smallest element forrowed by largest element
END
```

إيجاد القيمة المتوسطة لعناصر المصفوفة

```
Find_mean_of_elements
        set i to 1
        set sum to zero
        DOWHILE i <= number_of_elements
            sum = sum + array(i)
            i = i + 1
        ENDDO
        mean = sum/number_of_elements
        print mean
END
```

هياكل البيانات الديناميكية

يسمى هيكل بيانات مثل المصفوفات هيكل بيانات استاتيكي بمعنى أننا نحدد حجمها، أي العدد الأقصى لعناصرها، عند الإعلان عنها. أما هيكل البيانات الديناميكي فيتميز بأن حجمه يمكن أن يتسع أو يضيق بحسب متطلبات البرنامج. ويطلق على أضغر وحدة في مثل هذه الهياكل "لبنة **nodes**".

وفي إنشاء هياكل البيانات تستخدم "المؤشرات **pointers**" لإضافة لبنة جديدة، أو إلغاء لبنة موجودة، أو ربط اللبنات بعضها بالبعض. والمؤشر هو متغير يخزن فيه عنوان البيانات الأخرى في الذاكرة. فالمؤشر بذلك يستخدم كطريقة غير مباشرة للتعامل مع البيانات.

وسوف نعرض في القسم التالي بعض هياكل البيانات الديناميكية، وأمثلة للخوارزمات التي تتعامل معها.

الصفوف

الصف queue هو هيكل بيانات يعامل على أساس "الداخل أولا هو الخارج أولا first-in-first-out" [يختصر اسم هذا النظام FIFO]. ويمكن تشبهه بصفوف المتعاملين مع البنك، حيث يكون أو فرد في الصف هو أول من يصل للموظف المسئول.

ولدينا عمليتان يمكن أن تجريا على الصفوف؛ دخول الصف، بإضافة لبنة لمؤخرة الصف، أو الخروج من الصف، بقراءة اللبنة في مقدمة الصف ثم محوها منه.

وبعض اللغات البرمجية لا تتضمن تمثيل الصفوف، والحل هنا هو استخدام مصفوفة لتمثيل الصف، وتطبيق نظام المؤشرات عن طريق متغير له دليل يعبر عن موضع كل لبنة في الصف. كما نحتاج أيضا إلى مؤشرين لتحديد موضعي مقدمة الصف top اللبنة عليها الدور في الخروج من الصف] ومؤخرته tail، أي المكان الذي يمكن أن يضاف إليه اللبنة الجديدة، فلهذا التحديد أهميته في تعريف الصف برمجيا. وتصمم الكثير من الصفوف بنظام **الطي wrapping** بمعنى أن تطوى مقدمة الصف على مؤخرته عندما يكون ذلك مطلوبا.

ونستخدم المتغيرات التالية في التعامل مع الصفوف:

queue: اسم المصفوفة المعبرة عن الصف

max_size: العدد الأقصى لعناصر لصف

queue_counter: عدد اللبنات الفعلي في الصف

queue_head: مؤشر يشير لموضع مقدمة الصف، أي اللبنة التي عليها الدور في القراءة والمحو

queue_tail: مؤشر يشير لموضع مؤخرة الصف، أي المكان الذي يمكن أن تحتله اللبنة الجديدة.

وسوف نقدم خوارزمين لدخول الصف والخروج منه.

```
1-    Add_to_queue
      IF (queue_tail=queue_head AND queue_counter >0)
```

```
        Print errror message "queue overflow"
    ELSE
        queue(queue_ tail) = new item
        queue_ tail = queue_ tail+1
        IF queue_ tail > max_size
            queue_ tail = 1
        ENDIF
    queue_counter = queueu_counter +1
    ENDIF
END

2-  Remove_from_queue
    IF (queue_counter= 0) THEN
        Print errror message "queue empty"
    ELSE
        required value=queue(queue_ head)
        queue_ head = queue_ head+1
        IF queue_ head > max_size THEN
            queue_ head = 1
        ENDIF
    queue_counter = queueu_counter -1
    ENDIF
END
```

وتلاحظ أن اللبنة التي في المقدمة تخرج من الصف بمجرد أن يشير المؤشر queue_ head للبنة التالية كمقدمة للصف، وذلك بالأمر:

queue_ head = queue_ head+1

وتلاحظ أن اللبنة التي في المقدمة تخرج من الصف بمجرد أن يشير المؤشر queue_ head للبنة التالية كمقدمة للصف، وذلك بالأمر:

queue_ head = queue_ head+1

معنى ذلك أن المحو لا يتم ماديا، فالقيمة التي تخرج من الصف لا تمحى، بل تظل في موضعها، حيث إنها سوف يكتب عليها حين يأتي الدور على موضعها في الإضافة.

المكادس

المكدس **stack** هو هيكل للبيانات يعالج بنظام الداخل أخيرا هو الخارج أولا -last-in-first out [يسمى LIFO]، ويمكن تشبيهه برصة الأطباق في المقصف، فآخر طبق يدخل في الرصة، والذي يوضع على قمتها، هو الأول الذي يستخدم، كما أن الطبق الجديد يوضع أيضا علـــى القمة. وغالبا ما يطلق على عمليا الإضافة للمكدس **push** وللمحو منها **pop**[i].

ومرة أخرى، فإن أيسر وسيلة للتعامل مع المكادس هو تصويرها على هيئة مصفوفة. وســـوف نستخدم في الخوارزمين التاليين المتغيرات التالية:

stack: المصفوفة المعبرة عن المكدس

max_size: العدد الأقصى لعناصر المكدس

top_of_stack: مؤشر يشير لموضع قمة المكدس

```
1-      Add_to_stack (push)
        IF top_of_stack NOT = max_size THEN
              top_of_stack = top_of_stack +1
              stack(top_of_stack) = new item
        ELSE
              Print error message "Stack overflow"
        ENDIF
END
1-      Remove_from_stack (pop)
        IF top_of_stack NOT = zero THEN
              value required= stack(top_of_stack)
              top_of_stack = top_of_stack +1
        ELSE
              Print error message "Stack underflow"
        ENDIF
END
```

وهنا أيضا لا يكون المحو ماديا.

القوائم المترابطة

القوائم المترابطة **linked lists** هي هيكل للبيانات تضم سلسلة من الخلايا، تحتوي كل خلية على بيان ما، ومؤشر في نهايتها يشير للخلية التالية، كما هو موضح في الشكل التالي. والشكل لنوع من هذه القوائم يسمى "القوائم الخطية **linear lists**":



وفي الشكل المبين، يشير المؤشر المسمى **first** إلى مقدمة القائمة، أي أول خلية فيها، والمؤشر المسمى **current** إلى الخلية الحالية، أما المؤشر **null** في آخر خلية فيحدد نهاية القائمة.

وفائدة هذا النوع من هياكل البيانات أن التحكم في ترتيب البيانات يمكن أن يتم بالتغيير في قيم المؤشرات دون الاضطرار إلى المساس بالوضع المادي لها، فيكون الترتيب المنطقي للخلايا غير مرتبط بالترتيب المادي لها.

وحينما لا تكون لغة البرمجة مصممة للتعامل المباشر مع القوائم المترابطة، فإن تمثيلها يكون على باستخدام "**المصفوفات المتوازية parallel arrays**". وهنا، سوف نستخدم مصفوفتين، الأولى تتضمن عناصر القائمة، أما الثانية، فتتضمن "الروابط **links**"، أي تضم مقابل كل عنصر في المصفوفة الأولى ما يشير العنصر التالي له من وجهة الترتيب المنطقي المتبع. وبمعنى آخر، يمكن أن تتضمن المصفوفة التالية الرقم السفلي **subscript** للعنصر التالي. ونحتاج بداهة لمتغير يشير إلى موضع الخلية الأولى، كما نحتاج إلى قيمة معينة، تخرج على الترتيب الموضوع، يتفق على أنها تعني نهاية القائمة **null**.

وفي الخوارزمات التالية سوف نستخدم المسميات التالية:

items: مصفوفة العناصر

links: مصفوفة الروابط

first: متغير المؤشر للخلية الأولى

current: متغير المؤشر للخلية الحالية

last: متغير المؤشر للخلية السابقة

continue: متغير منطقي يوضع في حالة التحقق true حينما يراد الاستمرار في عملية البحث

والأمثلة التالية تتعامل مع القوائم الخطية المفردة singly linked linear lists:

```
1       Traverse_and_print طباعة محتويات القائمة
        current = first
        DOWHILE current NOT = null
                Print intem(current)
                current = links(current)
        ENDDO
END
2       Search_for_value
                current = first
                contiue = true
                DOWHILE continue AND (current NOT     null)
                        IF items(current) NOT = value THEN
                                last = current
                                current = links(current)
                        ELSE
                                contiue = false
                        ENDIF
                ENDDO
        END
```

وتلاحظ أن المتغير continue سوف يظل متحققا إذا لم يتم العثور على المتغير المنشود، أي إذا وصلت قيمة current إلى null.

يفترض الخوارزم التالي أن القيمة المطلوب البحث عنها في المثال السابق قد وجدت، وأنه يسراد محوها.

```
3-      Remove_from_list
        IF NOT continue THEN إذا وجدت القيمة، إذن
                IF current = first THEN
                        first = links(current)
```

```
            ELSE
                  links(last) = links(current)
            ENDIF
      ENDIF
END
```

وفي التطبيقات العملية يسجل الموضع الذي تم إخلاؤه لكي يستخدم فيما بعد، ويحتاج ذلك إلى متغير يسمى free لكي يحتفظ فيه بقيمة الموضع الخالي. ويمكن بعد الانتهاء من عملية المحو تحديث قائمة المواضع الشاغرة بالأمرين:

```
links(current) = free
free = current
```

وهناك العديد من صور القوائم المترابطة، مثل الشجرات الثنائية binary trees والأشكال الرسومية graphs، وهو ما يخرج عن نطاق هذا الكتاب.

ملحق 3 - مسرد المصطلحات

ظ = انظر، قا: قارن

إخفاء البيانات data hiding في التصميم الكائني، تكون بيانات الكائنات مخفية عن بعضها البعض.

بلوك block مجموعة من تعليمات البرنامج تعمل كوحدة واحدة، كجزء تكـــراري في دوارة مثلا.

بيانات أولية elementary data المتغيرات والثوابت، والتي تعامل مســـتقلة أو مجمعـــة في هياكل بيانات (ظ)

تحليل بنائي modularisation تحليل البرنامج إلى وحدات بنائية (ظ).

تحليل وظيفي functioanl analysis = تصميم بنائي

ترابط cohesion معيار لقياس قوة البنية الداخلية لوحدة بنائية، أي إلى أي مـــدى تكـــون التعليمات أو العناصر المكونة منها الوحدة وثيقة العلاقة فيما بينها، وكلما زاد الترابط كـــانت الوحدة أفضل تصميما.

تصميم بنائي modular design تصميم البرامج بناء على التحليل البنائي.

تصميم كائني object oriented design تصميم ينظر للبرنامج كعلاقات بين الكائنـــات وليس بين الدوال.

تقارن coupling قوة الاتصال بين الوحدات البنائية للبرنامج، وكلما كـــان التقـــارن بـــين الوحدات ضعيفا، دل ذلك على قوة استقلالية الوحدات، وكان التصميم أفضل.

ثابت cons ant:بيان مخزن في الذاكرة كقيمة ثابتة لا تتغير طوال تنفيذ البرنامج (قا: متغير)

جدول تحليلي defining diagram جدول يحلل فيه توصيف البرنـــامج إلى مدخـــلات، عمليات معالجة ومخرجات، تمهيدا لصياغة خوارزم الحل.

حرف lette الحروف الأبجدية (قا: محرف)

خارطة هيكلية heirarchy chart شكل يبين التدرج الهرمي للوحدات البنائية للبرنـــــامج على صورة هيكل تنظيمي أشبه بالهياكل التنظيمية في المنشآت.

خوارزم algorithm مجموعة من الأوامر المنظمة والمفصلة والواضحة، توضع لكي تصـــف العمليات الضرورية لإنتاج مخرجات معينة من مدخلات معينة.

دالة رئيسية mainline الدالة الأساسية في البرنامج التي تتحكم في سير البرنامج، وهي الـــتي تتولى استدعاء الدوال الأخرى كل في دوره.

صف queue هيكل بيانات يعمل على أساس البيان الذي يدخل أولا first in يخـــرج أولا first out (FIFO) (قا: مكدس)

قائمة مترابطة linked list نوع من هياكل البيانات مكون من تسلسل من الخلايا، كل خلية تضم بيانا، ومؤشرا يشير للخلية التالية.

كائن object تجميع البيانات مع الدوال التي تعمل عليها في وحدة واحدة مستقلة

كود code: 1) في البرمجة: أ- صياغة البرنامج في لغة برمجية، ب- جزء من برنامج مصاغ بلغة برمجية 2) في علمي المعلوماتية والاتصالات، وضع المعلومات في صورة متعارف عليها (مثـــلا: كود الآسكي)

لبنة node الوحدة الأساسية في هيكل البيانات والتي لا يمكن أن تجزأ إلى أبسط منها

مؤشر pointer متغير يحتوي على عنوان متغير آخر.

متغير variable مكان في الذاكرة تخزن فيه قيمة أحد بيانات البرنامج

متغير بولي (منطقي) Boolean (logical) variable: متغير له حالتان فقط، إما صحيـــح أو خطأ

متغير عام global variable متغير متاح التعامل معه لكل دوال البرنامج

متغير محلي local variable متغير غير متاح التعامل معه إلا للدالة المعرف بها

محرف character كل ما يدخل من لوحة المفاتيح من حروف أبجدية وعلامات خاصة، وأيضا الأرقام إذا لم تكن داخلة في عمليات حسابية، كالأرقام الداخلة في الأكواد مثلا (قـا: حرف)

مصفوفة array: هيكل للبيانات مكون من عدد من البيانات لها نفس النوع ويجمعها اسم مشترك

مكدس srack. هيكل بيانات يعامل على أساس البيان الذي يدخل أخيرا **last in** يخرج أولا **first out** (LIFO) (قا: صف)

هيكل التحكم CASE : هيكل يوسع من هيكل الاختيار الأساسي ليكون من عدة بدائل بدلا من بديلين فقط

هيكل بيانات data structure تجميع من أنواع أولية من البيانات، (ظ: مكدس، مصفوفة، قائمة مترابطة، صف، كائن)

هيكل حاكم control structure تنص النظرية الهيكلية في تصميم البرامج على أن البرنامج يصاغ بثلاثة أنواع من الهياكل التي تتحكم في تنفيذ البرنامج: الأوامر المتتابعة، الاختيار، الدوارات.

وحدة بنائية module جزء من البرنامج يمثل وظيفة مستقلة، ويطلق عليها في اللغات البرمجية: دالة، إجراء، برنامج فرعي.

ملحق 4- قاموس مصطلحات لاتيني-عربي

المصطلح	معناه
accumulator	مركم
algorithm	خوارزم
array	مصفوفة
balanced line algorithm	خوارزم الخط المتوازن
Boolean variable = logical variable	
character	محرف
code	كود
cohesion	ترابط داخلي
compiling	ترجمة البرنامج
constant	ثابت
control structure	هيكل حاكم
counter	عداد
coupling	تقارن
data structure	هيكل بيانات
debugging	تصحيح البرنامج
dynamic data structure	هيكل بيانات ديناميكي (قا: استاتيكي)

المصطلح	معناه
static data structure	هيكل بيانات استاتيكي (قا: ديناميكي)
flow charts	مخططات التدفق
function	دالة، وظيفة
elementary data type	نوع بيانات أولي
functional decomposition	تحليل وظيفي
global data	بيان عام (قا: محلي)
hierarchy chart	خارطة هيكلية
information hiding	إخفاء البيانات
increment	تزايد (زيادة بمقدار الوحدة)
infinite loop	دوارة لانهائية
initialization	استهلال
interactive	تفاعلي
leading decision loop	دوارة متقدمة الاختبار
letter	حرف أبجدي
linked list	قائمة مترابطة
literal	ثابت محرفي
local data	بيان محلي
loop	دوارة
mainline	الدالة الرئيسية
modular design	تصميم بنائي
modularisation	تحليل بنائي

معناه	المصطلح
وحدة بنائية	module
كائن	object
تصميم كائني	object oriented design
معامل	parameter
مؤشر	pointer
القراءة التمهيدية	priming read
لغة شبه برمجية	pseudocode = pseudolanguage
صف	queue
سجل	record
مدى	scope
السجل الرقيب	sentinel
مكدس (ج: مكادس)	stack
عبارة نصية	string
لغة مهيكلة	structrured language
البرمجة المهيكلة	structured programming
وحدة طرفية (مكونة من لوحة مفاتيح وشاشة)	terminal
سجل التذييل	trail record
دوارة متأخرة الاختبار	trailing decision loop
متغير	variable

خاتمة

الهدف الأساسي من هذا الكتاب هو تشجيع المبرمجين على اتباع خطوات منهجية بسيطة تؤدي بهم إلى تصميمات جيدة لبرامجهم. هذه الخطوات هي:

1. تحليل المسألة: للقيام بذلك، يجري وضع خطوط تحت الأسماء والصفات، ثم الأفعال والظروف بشكل مغاير. ويساعد ذلك على التمييز بين العناصر الرئيسية؛ المدخلات والمخرجات وعمليات المعالجة المطلوبة. يصاغ هذا التحليل في جدول يطلق عليه "الجدول التحليلي" [في النص الإنجليزي: "شكل التعريف **defining diagram**"].

في هذه الخطوة عليك التركيز فكريا عما هو مطلوب تنفيذه، بأن تسجل ببساطة المهام المطلوب تنفيذها دون أن تشغل فكرك بكيفية التنفيذ [فلا تهتم مثلا بمعادلة إخراج القيم المطلوبة].

2. تعريف الدوال: تجمع العمليات في مجموعات تمثل مهام ذات وحدة منطقية، مستعينا بالجدول التحليلي. هذه المهام يطلق عليها "دوال **functions**". ويمكن أن تقسم المهام إلى مهام فرعية. بذلك تحدد الدوال الخاصة بالبرنامج.

وقد لا تكون كافة الأنشطة المنفذة متضمنة في الجدول التحليلي، ففي حالة البرامج الضخمة تجد أن الأنشطة العامة فقط هي المذكورة في مرحلة التحليل. والهدف من انتهاج أسلوب التصميم من أعلى إلى أسفل **top-down design** هو أن يكون البدء بالمهام الأكثر عمومية، وأن ينظر في الأنشطة الأدنى مرتبة فقط عند الانتهاء من وضع الأنشطة الأعم. فعليك التركيز على المهام الأساسية قبل أن تفكر في المهام الفرعية.

3 . وضع خريطة هيكلية: تجمع كافة الدوال بمستوياتها المختلفة على شكل تصويري في خريطة تأخذ الدوال فيها شكل الوحدات البنائية **modules**، تبين التدرج الهرمي لها والعلاقة فيما بينها. وتشبه هذه الخريطة الخرائط الهيكلية التي تبين التنظيم الإداري في المنشآت.

وبالضبط كما في المنشآت، حيث يقوم المدير بتغيير الهيكل للوصول إلى أفضل وضع ملائم، عليك أن تفعل نفس الشيء إلى أن ترضى تماما عن تنظيمك الهيكلي. ومن الممارسات المفيدة في البرمجة دراسة الخريطة الهيكلية جيدا، ومحاولة الوصول إلى أفضل شكل لها أن تكون أبسط ما يمكن، وتعبر عن أفضل تدرج هرمي.

ولنتذكر أنك إلى هذه المرحلة مهتم بما يراد عمله وليس بطريقة التنفيذ. وما أن تنتهي من وضع الخريطة الهيكلية حتى يكون الأوان قد حان للتفكير في المنطق العام لسير البرنامج.

4 . ضع المنطق العام للدالة الأساسية **mainline**: ويصاغ خوارزم هذا المنطق باللغة شبه البرمجية **psuedocode**، وهي صورة من اللغة الإنجليزية الطبيعية وضعت في هيئة ملائمة لتحويلها فيما بعد إلى لغة برمجية، وتستخدم فيها الكلمات الحاكمة والركيزية **key words** وكذا الإزاحة **indentations** لتعكس الهياكل المستخدمة في البرمجة: الأوامر المتتابعة **sequences** والدوارات **loops** والقرارات **decisions** [العبارات الشرطية].

وتتضمن الدالة الأساسية الدوال الرئيسية [دوال المستوى الأول] وتتابع تنفيذها. وقد رأينا أن أغلب الدوال الرئيسية تتبع نمطا واحدا، عمليات استهلالية **initialization proceedings** (افتتاحية) ثم دوارة تتضمن عمليات معالجة لسجلات، ثم بعض العمليات الختامية.

5. وضع خوارزمات الدوال الفرعية: ويكون ذلك بنفس المنهج العام، التصميم من أعلى إلى أسفل، أي بوضع الخوارزم للوظيفة الأساسية قبل تجزئتها إلى وظائف فرعية. ويعتبر التصميم البنائي للبرنامج ككل قد تم عند الانتهاء من وضع خوارزمات الدوال الأدنى مستوى.

6. الاختبار المكتبي للدوال: يقصد بذلك تتبع سير الخوارزم طبقا لمتطلبات البرنامج كما وردت في وصفه العام، للتحقق من أنه سوف يصل بالفعل للنتائج المرجوة. ويكون ذلك باختيار بيانات تجريبية بسيطة، وتصور المخرجات التي يفترض أن يحققها البرنامج من هذه المدخلات. بعد ذلك يجرى تتبع لأوامر البرنامج على المدخلات كما وردت في الخوارزم، وتقارن نتيجته النهائية بما يفترض أن يحققه البرنامج.

وبهذه الطريقة يمكن اكتشاف الأخطاء المنطقية **logical errors** التي كثيرا ما تحدث في صياغة البرامج، ولا تكتشفها عمليات التصحيح **debugging**.

إضافة من المترجم

الأخطاء المنطقية وأخطاء الصياغة:

تقوم عمليات التصحيح للبرامج لاكتشاف أخطاء الصياغة syntax errors في وضع البرامج، ولذا فهي منصبة على صحة صياغة أوامر اللغة البرمجية ذاتها، كنسيان الفاصلة المنقوطة "؛" في لغة السي. وأخطاء الصياغة تمنع تشغيل البرنامج أصلا، حيث لا يقبل الترجمة compiling.

أما الأخطاء المنطقية فتتضح حين يشغل البرنامج فيؤدي لنتائج غير المفترض حدوثها. وليس من سبيل لاكتشافها غير مراجعة خوارزم البرنامج، وبتوقف نجاح تصحيحها على خبرة المراجع في منطق البرنامج.

ويمكن ضرب المثال التالي لتوضيح الفرق:

تصور أنك أصدرت الأمر التالي لخادمك: "اذهب من السوق واشتر فولا". فإذا كان خادمك حساس لغويا، كما هو الحال في اللغات البرمجية، فلن يقبل منك هذا الأمر، لخطأ استخدامك حرف الجر.

أما لو أصدرت إليه الأمر: "اذهب للسوق واشتر فيلا" وكان له مثل ذكاء حاسوبك، فسوف ينهض للتنفيذ مهما كلفه البحث عن المطلوب (الفيل) من عناء، فهو غير واع لمجافاة الأمر للمنطق، ولن يعود إليك إلا ببحثك عنه وإرجاعه (بلغة البرمجة يمكن القول بأنه قد دخل في دوارة لانهائية infinite loop، حيث لا كون الخروج منها إلا بإعادة تشغيل الجهاز).

رقم الإيداع
1998 / 2860
I.S.B.N
977-5603-19-6

يتناول هذا الكتاب موضوعا لا غنى عنه لدارسي البرمجة، وهو تصميم البرامج الحاسوبية. وتقدم المؤلفة فيه منهجا مبسطا لوضع تصميمات قوية خالية من الأخطاء المنطقية بقدر الإمكان. والمؤلفة وهي بسبيلها لتقديم هذا المنهج، تقوم بخدمتين إضافيتين لدارسي البرمجة، أولهما أنها تعرض لمفاهيم البرمجة الأساسية بالشرح الوافي، وثانيهما أنها تعزز الشرح بأمثلة تغطي عمليا كافة المواقف البرمجية [illegible]

وفي اعتقادنا أننا بتقديم هذا الكتاب لأبنائنا من دارسي البرمجة نكون قد [illegible] ثغرة خطيرة في المكتبة العربية، والتي تخلو أساسا من مراجع متخصصة في مجال علوم الحاسبات عموما.

وإتماما للفائدة قدمنا في نهاية الكتاب مسردا بالمصطلحات [illegible] المدخل العربي يشمل تعريفا مختصرا لها، ومقابلها باللغة الإنجليزية، وقاموسا لأهم المصطلحات [illegible] المدخل اللاتيني

المهندس / علي يوسف

خوارزم للنشر والتوزيع و الكمبيوتر
خلف ٤٠ شارع بورسعيد - الإسكندرية
☎ ٤٨٣٦١٨٦